# React.js Notes – Part 2

Contents

[React.js Notes – Part 2 1](#_Toc113548923)

[Section 5: Rendering Lists & Conditional Content 1](#_Toc113548924)

[63. Module Introduction 1](#_Toc113548925)

[64. Rendering Lists of Data 2](#_Toc113548926)

[65. Using Stateful Lists 30](#_Toc113548927)

[66. Understanding "Keys" 36](#_Toc113548928)

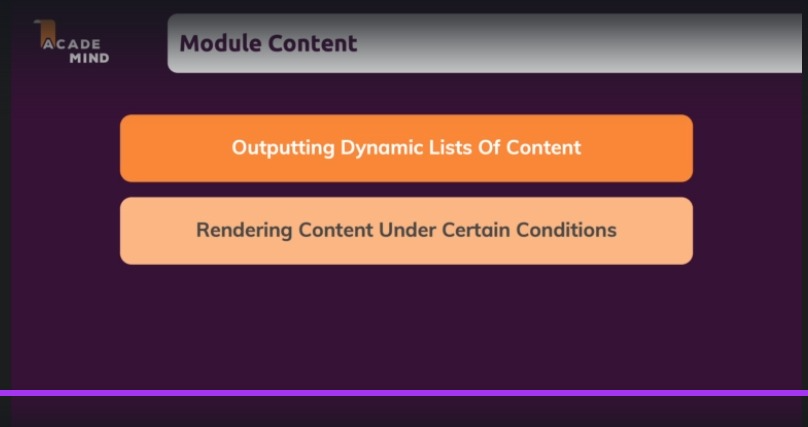
## Section 5: Rendering Lists & Conditional Content

### 63. Module Introduction

In this section, we will dive into rendering lists and conditional content as well as working with really dynamic content. We will look at how we can output arrays of data on our page, and how we can show different content based on different conditions.



This is what we will look at in this module. We will continue working on the application we already started over the last course sections, but in this module, we will finish it by adding these features.



### 64. Rendering Lists of Data

Our list is static currently, not dynamic, meaning that as we add a new item to the form, it's not added to the list.

We want to render our expenses which are defined in App.js. We have an expenses array. We want to render this in the Expenses component. Therefore, the first step is to pass our expenses down via props, which we are already doing. We are passing the items prop,

|  |
| --- |
| src/App.js |
| import React from "react";  import NewExpense from "./components/NewExpense/NewExpense";  import Expenses from "./components/Expenses/Expenses";  const App = () => {  const expenses = [  {  id: "e1",  title: "Toilet Paper",  amount: 94.12,  date: new Date(2020, 7, 14),  },  { id: "e2", title: "New TV", amount: 799.49, date: new Date(2021, 2, 12) },  {  id: "e3",  title: "Car Insurance",  amount: 294.67,  date: new Date(2021, 2, 28),  },  {  id: "e4",  title: "New Desk (Wooden)",  amount: 450,  date: new Date(2021, 5, 12),  },  ];  const addExpenseHandler = expense => {  console.log('In App.js');  console.log(expense);  }  return (  <div>  <NewExpense onAddExpense={addExpenseHandler}/>  <Expenses **items**={expenses} />  </div>  );  };  export default App; |

which points at this expenses array.

|  |
| --- |
| src/App.js |
| import React from "react";  import NewExpense from "./components/NewExpense/NewExpense";  import Expenses from "./components/Expenses/Expenses";  const App = () => {  const expenses = [  {  id: "e1",  title: "Toilet Paper",  amount: 94.12,  date: new Date(2020, 7, 14),  },  { id: "e2", title: "New TV", amount: 799.49, date: new Date(2021, 2, 12) },  {  id: "e3",  title: "Car Insurance",  amount: 294.67,  date: new Date(2021, 2, 28),  },  {  id: "e4",  title: "New Desk (Wooden)",  amount: 450,  date: new Date(2021, 5, 12),  },  ];  const addExpenseHandler = expense => {  console.log('In App.js');  console.log(expense);  }  return (  <div>  <NewExpense onAddExpense={addExpenseHandler}/>  <Expenses items={**expenses**} />  </div>  );  };  export default App; |

So, in the Expenses component, we do get our list of expenses, but we are currently not using that, and that is what we want to change.

|  |
| --- |
| src/components/Expenses/Expenses.js |
| import React, {useState} from "react";  import ExpenseItem from "./ExpenseItem";  import ExpensesFilter from "./ExpensesFilter";  import Card from "../UI/Card";  import "./Expenses.css";  const Expenses = (props) => {  const [filteredYear, setFilteredYear] = useState('2020');  const filterChangeHandler = selectedYear => {  setFilteredYear(selectedYear);  }  return (  <div>  <Card className="expenses">  <ExpensesFilter selected={filteredYear} onChangeFilter={filterChangeHandler}/>  <ExpenseItem  title={props.items[0].title}  amount={props.items[0].amount}  date={props.items[0].date}  />  <ExpenseItem  title={props.items[1].title}  amount={props.items[1].amount}  date={props.items[1].date}  />  <ExpenseItem  title={props.items[2].title}  amount={props.items[2].amount}  date={props.items[2].date}  />  <ExpenseItem  title={props.items[3].title}  amount={props.items[3].amount}  date={props.items[3].date}  />  </Card>  </div>  );  };  export default Expenses; |

On our props, we have this items prop and the value of that will be that array,

|  |
| --- |
| src/App.js |
| import React from "react";  import NewExpense from "./components/NewExpense/NewExpense";  import Expenses from "./components/Expenses/Expenses";  const App = () => {  const expenses = [  {  id: "e1",  title: "Toilet Paper",  amount: 94.12,  date: new Date(2020, 7, 14),  },  { id: "e2", title: "New TV", amount: 799.49, date: new Date(2021, 2, 12) },  {  id: "e3",  title: "Car Insurance",  amount: 294.67,  date: new Date(2021, 2, 28),  },  {  id: "e4",  title: "New Desk (Wooden)",  amount: 450,  date: new Date(2021, 5, 12),  },  ];  const addExpenseHandler = expense => {  console.log('In App.js');  console.log(expense);  }  return (  <div>  <NewExpense onAddExpense={addExpenseHandler}/>  <Expenses **items**={expenses} />  </div>  );  };  export default App; |

which we in the end want to render here. But we don't want to output the array as text or anything like that.

|  |
| --- |
| src/components/Expenses/Expenses.js |
| import React, {useState} from "react";  import ExpenseItem from "./ExpenseItem";  import ExpensesFilter from "./ExpensesFilter";  import Card from "../UI/Card";  import "./Expenses.css";  const Expenses = (props) => {  const [filteredYear, setFilteredYear] = useState('2020');  const filterChangeHandler = selectedYear => {  setFilteredYear(selectedYear);  }  return (  <div>  <Card className="expenses">  <ExpensesFilter selected={filteredYear} onChangeFilter={filterChangeHandler}/>  **<ExpenseItem**  **title={props.items[0].title}**  **amount={props.items[0].amount}**  **date={props.items[0].date}**  **/>**  **<ExpenseItem**  **title={props.items[1].title}**  **amount={props.items[1].amount}**  **date={props.items[1].date}**  **/>**  **<ExpenseItem**  **title={props.items[2].title}**  **amount={props.items[2].amount}**  **date={props.items[2].date}**  **/>**  **<ExpenseItem**  **title={props.items[3].title}**  **amount={props.items[3].amount}**  **date={props.items[3].date}**  **/>**  </Card>  </div>  );  };  export default Expenses; |

Instead, we want to render one ExpenseItem per element in the array, and that's straightforward to do.

For this, we first of all need single curly braces, opening and closing, because we want to execute a dynamic expression in our JSX code, and that is done with curly braces. We can execute JavaScript expressions inside of the curly braces.

|  |
| --- |
| src/components/Expenses/Expenses.js |
| import React, {useState} from "react";  import ExpenseItem from "./ExpenseItem";  import ExpensesFilter from "./ExpensesFilter";  import Card from "../UI/Card";  import "./Expenses.css";  const Expenses = (props) => {  const [filteredYear, setFilteredYear] = useState('2020');  const filterChangeHandler = selectedYear => {  setFilteredYear(selectedYear);  }  return (  <div>  <Card className="expenses">  <ExpensesFilter selected={filteredYear} onChangeFilter={filterChangeHandler}/>  **{}**  <ExpenseItem  title={props.items[0].title}  amount={props.items[0].amount}  date={props.items[0].date}  />  <ExpenseItem  title={props.items[1].title}  amount={props.items[1].amount}  date={props.items[1].date}  />  <ExpenseItem  title={props.items[2].title}  amount={props.items[2].amount}  date={props.items[2].date}  />  <ExpenseItem  title={props.items[3].title}  amount={props.items[3].amount}  date={props.items[3].date}  />  </Card>  </div>  );  };  export default Expenses; |

The expression which I do want to execute here is that when I reach out to my array of data, in this case the array of expenses, for every element in the array, we want to create such a JSX element.

|  |
| --- |
| src/components/Expenses/Expenses.js |
| import React, {useState} from "react";  import ExpenseItem from "./ExpenseItem";  import ExpensesFilter from "./ExpensesFilter";  import Card from "../UI/Card";  import "./Expenses.css";  const Expenses = (props) => {  const [filteredYear, setFilteredYear] = useState('2020');  const filterChangeHandler = selectedYear => {  setFilteredYear(selectedYear);  }  return (  <div>  <Card className="expenses">  <ExpensesFilter selected={filteredYear} onChangeFilter={filterChangeHandler}/>  {}  **<ExpenseItem**  **title={props.items[0].title}**  **amount={props.items[0].amount}**  **date={props.items[0].date}**  **/>**  <ExpenseItem  title={props.items[1].title}  amount={props.items[1].amount}  date={props.items[1].date}  />  <ExpenseItem  title={props.items[2].title}  amount={props.items[2].amount}  date={props.items[2].date}  />  <ExpenseItem  title={props.items[3].title}  amount={props.items[3].amount}  date={props.items[3].date}  />  </Card>  </div>  );  };  export default Expenses; |

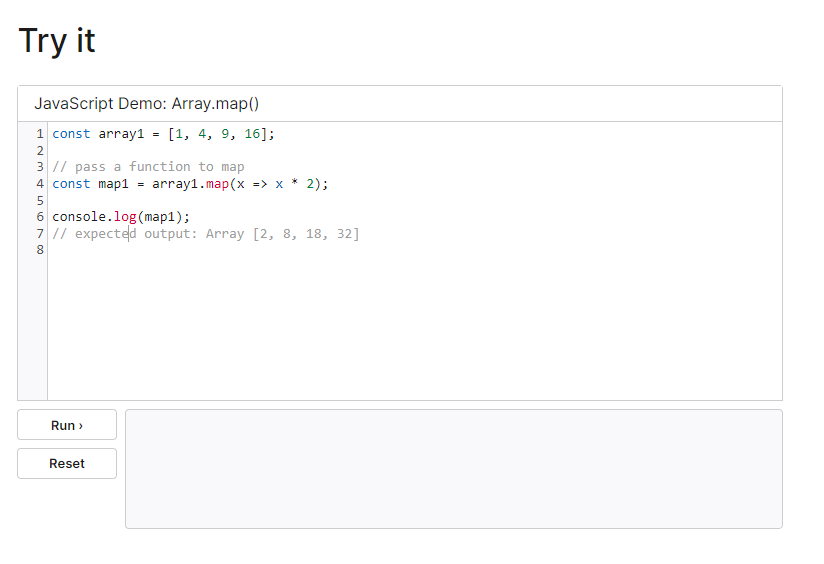
For this, we can access props.items (inside of the curly braces),

|  |
| --- |
| src/components/Expenses/Expenses.js |
| import React, {useState} from "react";  import ExpenseItem from "./ExpenseItem";  import ExpensesFilter from "./ExpensesFilter";  import Card from "../UI/Card";  import "./Expenses.css";  const Expenses = (props) => {  const [filteredYear, setFilteredYear] = useState('2020');  const filterChangeHandler = selectedYear => {  setFilteredYear(selectedYear);  }  return (  <div>  <Card className="expenses">  <ExpensesFilter selected={filteredYear} onChangeFilter={filterChangeHandler}/>  {**props.items**}  <ExpenseItem  title={props.items[0].title}  amount={props.items[0].amount}  date={props.items[0].date}  />  <ExpenseItem  title={props.items[1].title}  amount={props.items[1].amount}  date={props.items[1].date}  />  <ExpenseItem  title={props.items[2].title}  amount={props.items[2].amount}  date={props.items[2].date}  />  <ExpenseItem  title={props.items[3].title}  amount={props.items[3].amount}  date={props.items[3].date}  />  </Card>  </div>  );  };  export default Expenses; |

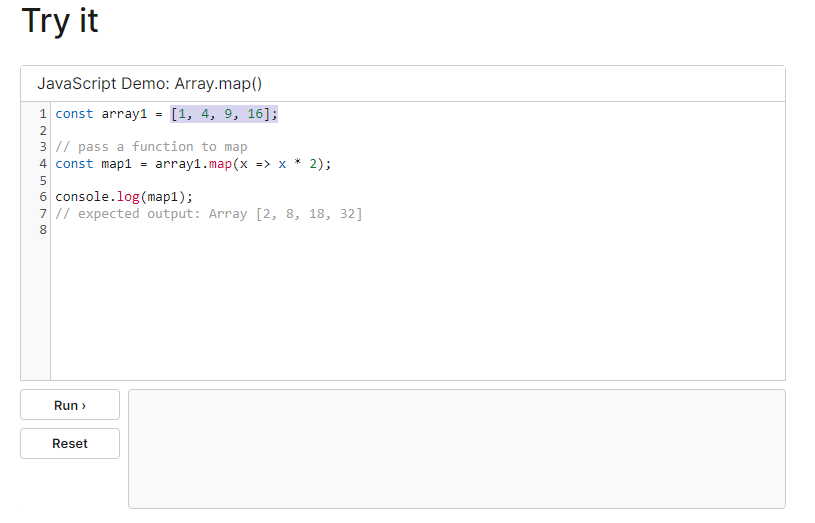
That's our array of items, our array of expenses, and now can use a built in array method, which is built into standard JavaScript, and that's the map method.

|  |
| --- |
| src/components/Expenses/Expenses.js |
| import React, {useState} from "react";  import ExpenseItem from "./ExpenseItem";  import ExpensesFilter from "./ExpensesFilter";  import Card from "../UI/Card";  import "./Expenses.css";  const Expenses = (props) => {  const [filteredYear, setFilteredYear] = useState('2020');  const filterChangeHandler = selectedYear => {  setFilteredYear(selectedYear);  }  return (  <div>  <Card className="expenses">  <ExpensesFilter selected={filteredYear} onChangeFilter={filterChangeHandler}/>  {props.items**.map()**}  <ExpenseItem  title={props.items[0].title}  amount={props.items[0].amount}  date={props.items[0].date}  />  <ExpenseItem  title={props.items[1].title}  amount={props.items[1].amount}  date={props.items[1].date}  />  <ExpenseItem  title={props.items[2].title}  amount={props.items[2].amount}  date={props.items[2].date}  />  <ExpenseItem  title={props.items[3].title}  amount={props.items[3].amount}  date={props.items[3].date}  />  </Card>  </div>  );  };  export default Expenses; |

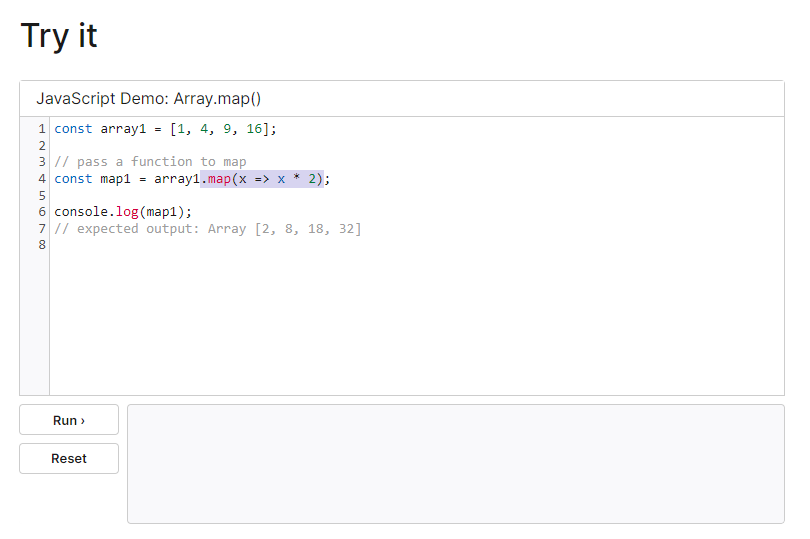
The map() method creates a new array based on another array, and that basically transforms every element in that original array. If we look at some [documentation for the map() method](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map), we see the following example:



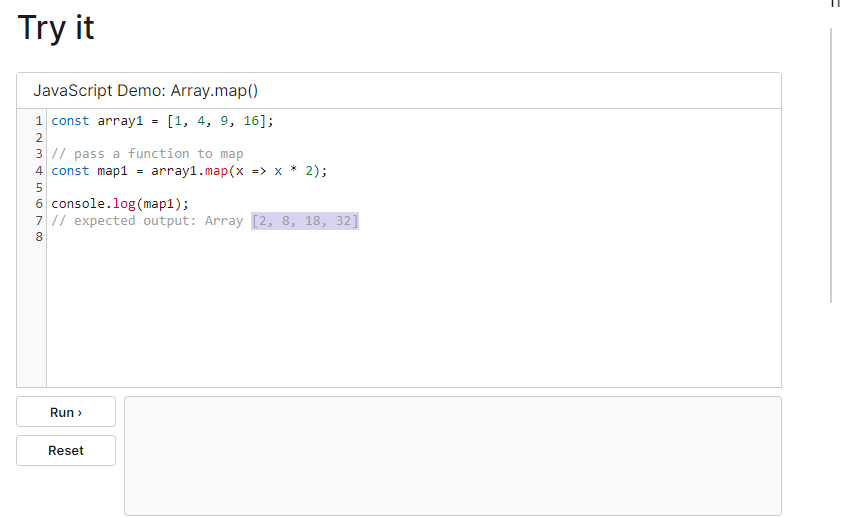
We have an array of numbers,



and with map(), we multiply every number with two,



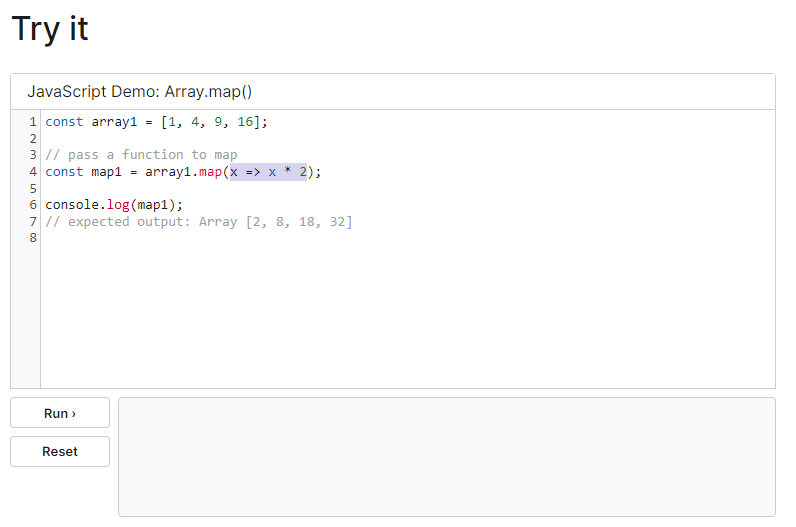
so that we get this new array as a result.



For applying this transformation, map(), this method which we can call,

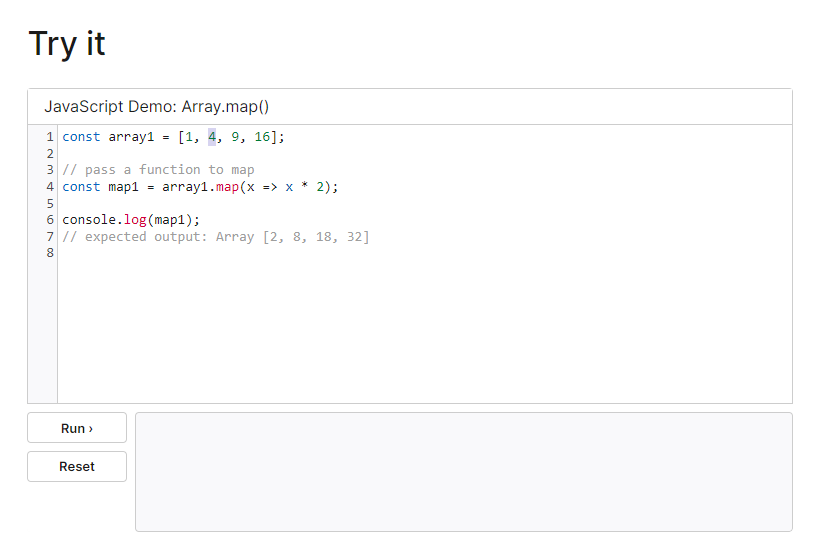


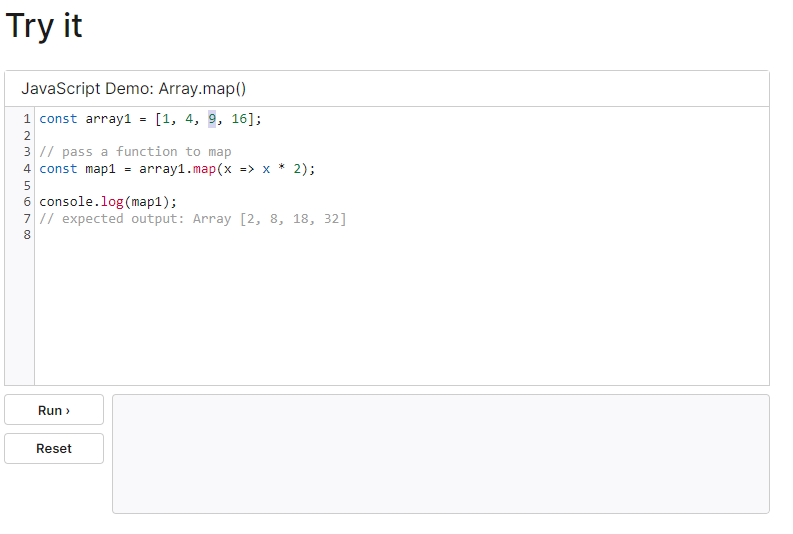
takes a function, which we pass as an argument, and that function is then executed

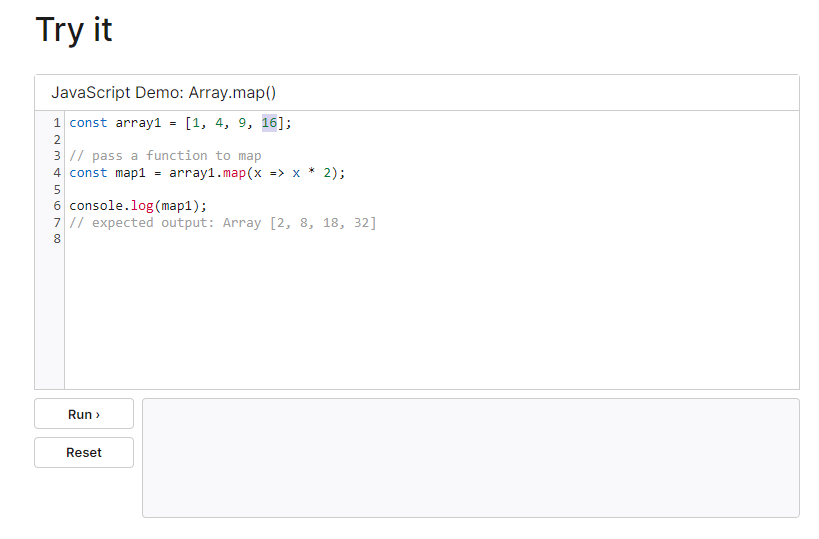


for every item in the array on which we're





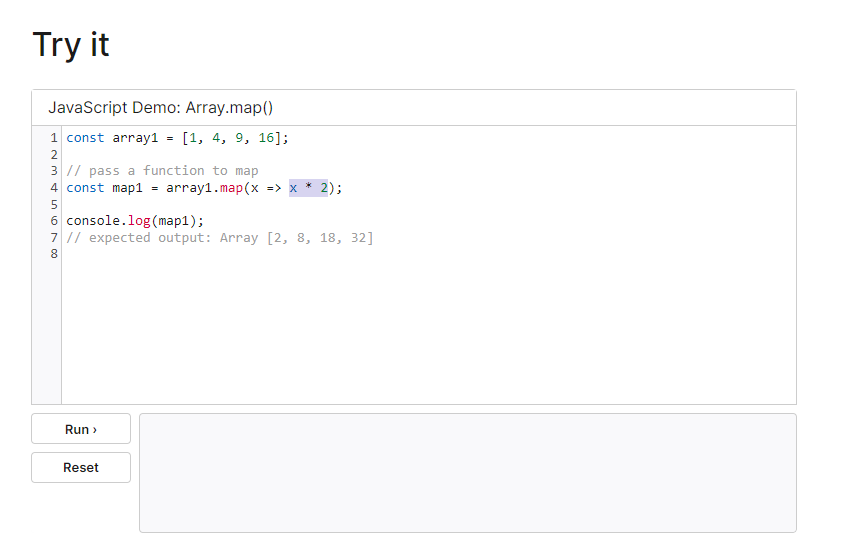




calling map(),



and the result of this function is the element which will be added to the newly created array.



That's how the map() method works. It's a built-in method, and we can utilize it in the Expenses component to transform our array of objects, which we have here,

|  |
| --- |
| src/App.js |
| import React from "react";  import NewExpense from "./components/NewExpense/NewExpense";  import Expenses from "./components/Expenses/Expenses";  const App = () => {  const expenses = **[**  **{**  **id: "e1",**  **title: "Toilet Paper",**  **amount: 94.12,**  **date: new Date(2020, 7, 14),**  **},**  **{ id: "e2", title: "New TV", amount: 799.49, date: new Date(2021, 2, 12) },**  **{**  **id: "e3",**  **title: "Car Insurance",**  **amount: 294.67,**  **date: new Date(2021, 2, 28),**  **},**  **{**  **id: "e4",**  **title: "New Desk (Wooden)",**  **amount: 450,**  **date: new Date(2021, 5, 12),**  **},**  **]**;  const addExpenseHandler = expense => {  console.log('In App.js');  console.log(expense);  }  return (  <div>  <NewExpense onAddExpense={addExpenseHandler}/>  <Expenses items={expenses} />  </div>  );  };  export default App; |

into an array of JSX elements, to be precise into an array full of ExpenseItem JSX elements.

|  |
| --- |
| src/components/Expenses/Expenses.js |
| import React, {useState} from "react";  import ExpenseItem from "./ExpenseItem";  import ExpensesFilter from "./ExpensesFilter";  import Card from "../UI/Card";  import "./Expenses.css";  const Expenses = (props) => {  const [filteredYear, setFilteredYear] = useState('2020');  const filterChangeHandler = selectedYear => {  setFilteredYear(selectedYear);  }  return (  <div>  <Card className="expenses">  <ExpensesFilter selected={filteredYear} onChangeFilter={filterChangeHandler}/>  {props.items.map()}  **<ExpenseItem**  **title={props.items[0].title}**  **amount={props.items[0].amount}**  **date={props.items[0].date}**  **/>**  <ExpenseItem  title={props.items[1].title}  amount={props.items[1].amount}  date={props.items[1].date}  />  <ExpenseItem  title={props.items[2].title}  amount={props.items[2].amount}  date={props.items[2].date}  />  <ExpenseItem  title={props.items[3].title}  amount={props.items[3].amount}  date={props.items[3].date}  />  </Card>  </div>  );  };  export default Expenses; |

If you do output an array of JSX elements here,

|  |
| --- |
| src/components/Expenses/Expenses.js |
| import React, {useState} from "react";  import ExpenseItem from "./ExpenseItem";  import ExpensesFilter from "./ExpensesFilter";  import Card from "../UI/Card";  import "./Expenses.css";  const Expenses = (props) => {  const [filteredYear, setFilteredYear] = useState('2020');  const filterChangeHandler = selectedYear => {  setFilteredYear(selectedYear);  }  return (  <div>  <Card className="expenses">  <ExpensesFilter selected={filteredYear} onChangeFilter={filterChangeHandler}/>  {props.items.map()}  **{}**  <ExpenseItem  title={props.items[0].title}  amount={props.items[0].amount}  date={props.items[0].date}  />  <ExpenseItem  title={props.items[1].title}  amount={props.items[1].amount}  date={props.items[1].date}  />  <ExpenseItem  title={props.items[2].title}  amount={props.items[2].amount}  date={props.items[2].date}  />  <ExpenseItem  title={props.items[3].title}  amount={props.items[3].amount}  date={props.items[3].date}  />  </Card>  </div>  );  };  export default Expenses; |

like a couple of Cards, then React is capable of simply rendering these elements. If you had something like this, an array of JSX elements as part of your JSX code, React would simply render these elements side by side.

|  |
| --- |
| src/components/Expenses/Expenses.js |
| import React, {useState} from "react";  import ExpenseItem from "./ExpenseItem";  import ExpensesFilter from "./ExpensesFilter";  import Card from "../UI/Card";  import "./Expenses.css";  const Expenses = (props) => {  const [filteredYear, setFilteredYear] = useState('2020');  const filterChangeHandler = selectedYear => {  setFilteredYear(selectedYear);  }  return (  <div>  <Card className="expenses">  <ExpensesFilter selected={filteredYear} onChangeFilter={filterChangeHandler}/>  {props.items.map()}  **{[<Card/>, <Card/>]}**  <ExpenseItem  title={props.items[0].title}  amount={props.items[0].amount}  date={props.items[0].date}  />  <ExpenseItem  title={props.items[1].title}  amount={props.items[1].amount}  date={props.items[1].date}  />  <ExpenseItem  title={props.items[2].title}  amount={props.items[2].amount}  date={props.items[2].date}  />  <ExpenseItem  title={props.items[3].title}  amount={props.items[3].amount}  date={props.items[3].date}  />  </Card>  </div>  );  };  export default Expenses; |

And that's what we utilize here to transform our array of objects into such an array of ExpenseItems, which then are rendered by React.

|  |
| --- |
| src/components/Expenses/Expenses.js |
| import React, {useState} from "react";  import ExpenseItem from "./ExpenseItem";  import ExpensesFilter from "./ExpensesFilter";  import Card from "../UI/Card";  import "./Expenses.css";  const Expenses = (props) => {  const [filteredYear, setFilteredYear] = useState('2020');  const filterChangeHandler = selectedYear => {  setFilteredYear(selectedYear);  }  return (  <div>  <Card className="expenses">  <ExpensesFilter selected={filteredYear} onChangeFilter={filterChangeHandler}/>  {props.items.map**()**}  <ExpenseItem  title={props.items[0].title}  amount={props.items[0].amount}  date={props.items[0].date}  />  <ExpenseItem  title={props.items[1].title}  amount={props.items[1].amount}  date={props.items[1].date}  />  <ExpenseItem  title={props.items[2].title}  amount={props.items[2].amount}  date={props.items[2].date}  />  <ExpenseItem  title={props.items[3].title}  amount={props.items[3].amount}  date={props.items[3].date}  />  </Card>  </div>  );  };  export default Expenses; |

So, map takes a function as an argument, and that function executes for every element in the array, and it gets that element for which it's currently executing as a parameter. So here we get our expense, for example, and then

|  |
| --- |
| src/components/Expenses/Expenses.js |
| import React, {useState} from "react";  import ExpenseItem from "./ExpenseItem";  import ExpensesFilter from "./ExpensesFilter";  import Card from "../UI/Card";  import "./Expenses.css";  const Expenses = (props) => {  const [filteredYear, setFilteredYear] = useState('2020');  const filterChangeHandler = selectedYear => {  setFilteredYear(selectedYear);  }  return (  <div>  <Card className="expenses">  <ExpensesFilter selected={filteredYear} onChangeFilter={filterChangeHandler}/>  {props.items.map(**expense** => )}  <ExpenseItem  title={props.items[0].title}  amount={props.items[0].amount}  date={props.items[0].date}  />  <ExpenseItem  title={props.items[1].title}  amount={props.items[1].amount}  date={props.items[1].date}  />  <ExpenseItem  title={props.items[2].title}  amount={props.items[2].amount}  date={props.items[2].date}  />  <ExpenseItem  title={props.items[3].title}  amount={props.items[3].amount}  date={props.items[3].date}  />  </Card>  </div>  );  };  export default Expenses; |

in the function body, so on the right side of the arrow, we then have to return the JSX element into which we want to map this expense. So, in this case, I want to map every expense in my expenses array into an ExpenseItem JSX element.

|  |
| --- |
| src/components/Expenses/Expenses.js |
| import React, {useState} from "react";  import ExpenseItem from "./ExpenseItem";  import ExpensesFilter from "./ExpensesFilter";  import Card from "../UI/Card";  import "./Expenses.css";  const Expenses = (props) => {  const [filteredYear, setFilteredYear] = useState('2020');  const filterChangeHandler = selectedYear => {  setFilteredYear(selectedYear);  }  return (  <div>  <Card className="expenses">  <ExpensesFilter selected={filteredYear} onChangeFilter={filterChangeHandler}/>  {props.items.map(expense => **<ExpenseItem />**)}  <ExpenseItem  title={props.items[0].title}  amount={props.items[0].amount}  date={props.items[0].date}  />  <ExpenseItem  title={props.items[1].title}  amount={props.items[1].amount}  date={props.items[1].date}  />  <ExpenseItem  title={props.items[2].title}  amount={props.items[2].amount}  date={props.items[2].date}  />  <ExpenseItem  title={props.items[3].title}  amount={props.items[3].amount}  date={props.items[3].date}  />  </Card>  </div>  );  };  export default Expenses; |

So I want to transform the expense object,

|  |
| --- |
| src/components/Expenses/Expenses.js |
| import React, {useState} from "react";  import ExpenseItem from "./ExpenseItem";  import ExpensesFilter from "./ExpensesFilter";  import Card from "../UI/Card";  import "./Expenses.css";  const Expenses = (props) => {  const [filteredYear, setFilteredYear] = useState('2020');  const filterChangeHandler = selectedYear => {  setFilteredYear(selectedYear);  }  return (  <div>  <Card className="expenses">  <ExpensesFilter selected={filteredYear} onChangeFilter={filterChangeHandler}/>  {props.items.map(**expense** => <ExpenseItem />)}  <ExpenseItem  title={props.items[0].title}  amount={props.items[0].amount}  date={props.items[0].date}  />  <ExpenseItem  title={props.items[1].title}  amount={props.items[1].amount}  date={props.items[1].date}  />  <ExpenseItem  title={props.items[2].title}  amount={props.items[2].amount}  date={props.items[2].date}  />  <ExpenseItem  title={props.items[3].title}  amount={props.items[3].amount}  date={props.items[3].date}  />  </Card>  </div>  );  };  export default Expenses; |

to this special kind of object, to this JSX element,

|  |
| --- |
| src/components/Expenses/Expenses.js |
| import React, {useState} from "react";  import ExpenseItem from "./ExpenseItem";  import ExpensesFilter from "./ExpensesFilter";  import Card from "../UI/Card";  import "./Expenses.css";  const Expenses = (props) => {  const [filteredYear, setFilteredYear] = useState('2020');  const filterChangeHandler = selectedYear => {  setFilteredYear(selectedYear);  }  return (  <div>  <Card className="expenses">  <ExpensesFilter selected={filteredYear} onChangeFilter={filterChangeHandler}/>  {props.items.map(expense => **<ExpenseItem />**)}  <ExpenseItem  title={props.items[0].title}  amount={props.items[0].amount}  date={props.items[0].date}  />  <ExpenseItem  title={props.items[1].title}  amount={props.items[1].amount}  date={props.items[1].date}  />  <ExpenseItem  title={props.items[2].title}  amount={props.items[2].amount}  date={props.items[2].date}  />  <ExpenseItem  title={props.items[3].title}  amount={props.items[3].amount}  date={props.items[3].date}  />  </Card>  </div>  );  };  export default Expenses; |

and then we can just configure this as we did it here.

|  |
| --- |
| src/components/Expenses/Expenses.js |
| import React, {useState} from "react";  import ExpenseItem from "./ExpenseItem";  import ExpensesFilter from "./ExpensesFilter";  import Card from "../UI/Card";  import "./Expenses.css";  const Expenses = (props) => {  const [filteredYear, setFilteredYear] = useState('2020');  const filterChangeHandler = selectedYear => {  setFilteredYear(selectedYear);  }  return (  <div>  <Card className="expenses">  <ExpensesFilter selected={filteredYear} onChangeFilter={filterChangeHandler}/>  {props.items.map(expense => <ExpenseItem />)}  **<ExpenseItem**  **title={props.items[0].title}**  **amount={props.items[0].amount}**  **date={props.items[0].date}**  **/>**  <ExpenseItem  title={props.items[1].title}  amount={props.items[1].amount}  date={props.items[1].date}  />  <ExpenseItem  title={props.items[2].title}  amount={props.items[2].amount}  date={props.items[2].date}  />  <ExpenseItem  title={props.items[3].title}  amount={props.items[3].amount}  date={props.items[3].date}  />  </Card>  </div>  );  };  export default Expenses; |

We add a title prop, but the title is now expense.title

|  |
| --- |
| src/components/Expenses/Expenses.js |
| import React, {useState} from "react";  import ExpenseItem from "./ExpenseItem";  import ExpensesFilter from "./ExpensesFilter";  import Card from "../UI/Card";  import "./Expenses.css";  const Expenses = (props) => {  const [filteredYear, setFilteredYear] = useState('2020');  const filterChangeHandler = selectedYear => {  setFilteredYear(selectedYear);  }  return (  <div>  <Card className="expenses">  <ExpensesFilter selected={filteredYear} onChangeFilter={filterChangeHandler}/>  {props.items.map(expense => <ExpenseItem **title={expense.title}** />)}  <ExpenseItem  title={props.items[0].title}  amount={props.items[0].amount}  date={props.items[0].date}  />  <ExpenseItem  title={props.items[1].title}  amount={props.items[1].amount}  date={props.items[1].date}  />  <ExpenseItem  title={props.items[2].title}  amount={props.items[2].amount}  date={props.items[2].date}  />  <ExpenseItem  title={props.items[3].title}  amount={props.items[3].amount}  date={props.items[3].date}  />  </Card>  </div>  );  };  export default Expenses; |

So, this expense, which is passed as a parameter

|  |
| --- |
| src/components/Expenses/Expenses.js |
| import React, {useState} from "react";  import ExpenseItem from "./ExpenseItem";  import ExpensesFilter from "./ExpensesFilter";  import Card from "../UI/Card";  import "./Expenses.css";  const Expenses = (props) => {  const [filteredYear, setFilteredYear] = useState('2020');  const filterChangeHandler = selectedYear => {  setFilteredYear(selectedYear);  }  return (  <div>  <Card className="expenses">  <ExpensesFilter selected={filteredYear} onChangeFilter={filterChangeHandler}/>  {props.items.map(**expense** => <ExpenseItem title={expense.title} />)}  <ExpenseItem  title={props.items[0].title}  amount={props.items[0].amount}  date={props.items[0].date}  />  <ExpenseItem  title={props.items[1].title}  amount={props.items[1].amount}  date={props.items[1].date}  />  <ExpenseItem  title={props.items[2].title}  amount={props.items[2].amount}  date={props.items[2].date}  />  <ExpenseItem  title={props.items[3].title}  amount={props.items[3].amount}  date={props.items[3].date}  />  </Card>  </div>  );  };  export default Expenses; |

into this function automatically, because that's how map works,

|  |
| --- |
| src/components/Expenses/Expenses.js |
| import React, {useState} from "react";  import ExpenseItem from "./ExpenseItem";  import ExpensesFilter from "./ExpensesFilter";  import Card from "../UI/Card";  import "./Expenses.css";  const Expenses = (props) => {  const [filteredYear, setFilteredYear] = useState('2020');  const filterChangeHandler = selectedYear => {  setFilteredYear(selectedYear);  }  return (  <div>  <Card className="expenses">  <ExpensesFilter selected={filteredYear} onChangeFilter={filterChangeHandler}/>  {props.items.map(**expense => <ExpenseItem title={expense.title} />**)}  <ExpenseItem  title={props.items[0].title}  amount={props.items[0].amount}  date={props.items[0].date}  />  <ExpenseItem  title={props.items[1].title}  amount={props.items[1].amount}  date={props.items[1].date}  />  <ExpenseItem  title={props.items[2].title}  amount={props.items[2].amount}  date={props.items[2].date}  />  <ExpenseItem  title={props.items[3].title}  amount={props.items[3].amount}  date={props.items[3].date}  />  </Card>  </div>  );  };  export default Expenses; |

that expense is used

|  |
| --- |
| src/components/Expenses/Expenses.js |
| import React, {useState} from "react";  import ExpenseItem from "./ExpenseItem";  import ExpensesFilter from "./ExpensesFilter";  import Card from "../UI/Card";  import "./Expenses.css";  const Expenses = (props) => {  const [filteredYear, setFilteredYear] = useState('2020');  const filterChangeHandler = selectedYear => {  setFilteredYear(selectedYear);  }  return (  <div>  <Card className="expenses">  <ExpensesFilter selected={filteredYear} onChangeFilter={filterChangeHandler}/>  {props.items.map(**expense** => <ExpenseItem title={expense.title} />)}  <ExpenseItem  title={props.items[0].title}  amount={props.items[0].amount}  date={props.items[0].date}  />  <ExpenseItem  title={props.items[1].title}  amount={props.items[1].amount}  date={props.items[1].date}  />  <ExpenseItem  title={props.items[2].title}  amount={props.items[2].amount}  date={props.items[2].date}  />  <ExpenseItem  title={props.items[3].title}  amount={props.items[3].amount}  date={props.items[3].date}  />  </Card>  </div>  );  };  export default Expenses; |

to extract the title.

|  |
| --- |
| src/components/Expenses/Expenses.js |
| import React, {useState} from "react";  import ExpenseItem from "./ExpenseItem";  import ExpensesFilter from "./ExpensesFilter";  import Card from "../UI/Card";  import "./Expenses.css";  const Expenses = (props) => {  const [filteredYear, setFilteredYear] = useState('2020');  const filterChangeHandler = selectedYear => {  setFilteredYear(selectedYear);  }  return (  <div>  <Card className="expenses">  <ExpensesFilter selected={filteredYear} onChangeFilter={filterChangeHandler}/>  {props.items.map(expense => <ExpenseItem title={**expense.title**} />)}  <ExpenseItem  title={props.items[0].title}  amount={props.items[0].amount}  date={props.items[0].date}  />  <ExpenseItem  title={props.items[1].title}  amount={props.items[1].amount}  date={props.items[1].date}  />  <ExpenseItem  title={props.items[2].title}  amount={props.items[2].amount}  date={props.items[2].date}  />  <ExpenseItem  title={props.items[3].title}  amount={props.items[3].amount}  date={props.items[3].date}  />  </Card>  </div>  );  };  export default Expenses; |

We can also use expense to extract the amount and date.

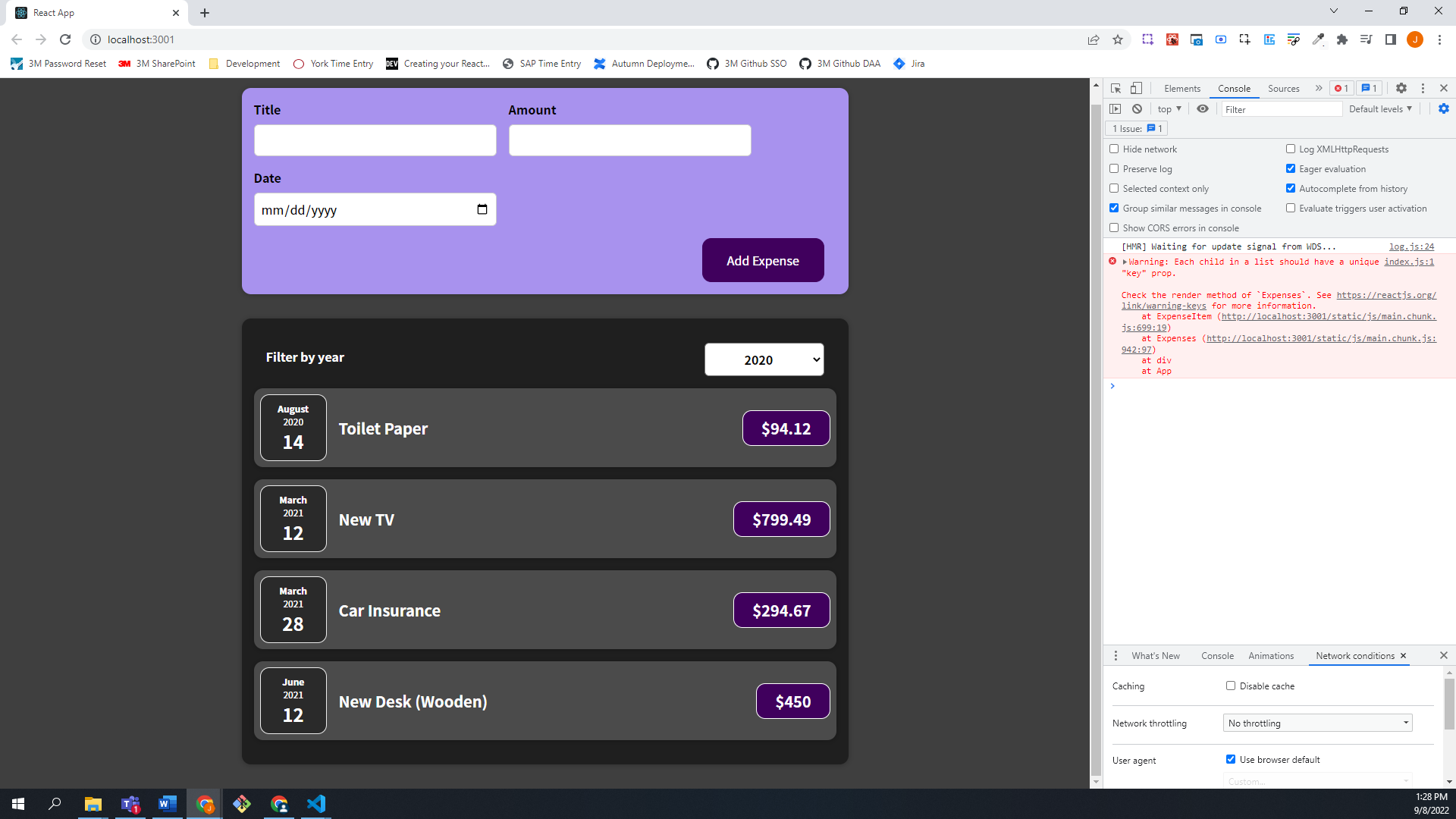
|  |
| --- |
| src/components/Expenses/Expenses.js |
| import React, { useState } from "react";  import ExpenseItem from "./ExpenseItem";  import ExpensesFilter from "./ExpensesFilter";  import Card from "../UI/Card";  import "./Expenses.css";  const Expenses = (props) => {  const [filteredYear, setFilteredYear] = useState("2020");  const filterChangeHandler = (selectedYear) => {  setFilteredYear(selectedYear);  };  return (  <div>  <Card className="expenses">  <ExpensesFilter  selected={filteredYear}  onChangeFilter={filterChangeHandler}  />  {props.items.map((expense) => (  <ExpenseItem  title={expense.title}  **amount={expense.amount}**  **date={expense.date}**  />  ))}  <ExpenseItem  title={props.items[0].title}  amount={props.items[0].amount}  date={props.items[0].date}  />  <ExpenseItem  title={props.items[1].title}  amount={props.items[1].amount}  date={props.items[1].date}  />  <ExpenseItem  title={props.items[2].title}  amount={props.items[2].amount}  date={props.items[2].date}  />  <ExpenseItem  title={props.items[3].title}  amount={props.items[3].amount}  date={props.items[3].date}  />  </Card>  </div>  );  };  export default Expenses; |

Now, we can get rid of the hard-coded ExpenseItems here.

|  |
| --- |
| src/components/Expenses/Expenses.js |
| import React, { useState } from "react";  import ExpenseItem from "./ExpenseItem";  import ExpensesFilter from "./ExpensesFilter";  import Card from "../UI/Card";  import "./Expenses.css";  const Expenses = (props) => {  const [filteredYear, setFilteredYear] = useState("2020");  const filterChangeHandler = (selectedYear) => {  setFilteredYear(selectedYear);  };  return (  <div>  <Card className="expenses">  <ExpensesFilter  selected={filteredYear}  onChangeFilter={filterChangeHandler}  />  {props.items.map((expense) => (  <ExpenseItem  title={expense.title}  amount={expense.amount}  date={expense.date}  />  ))}  ~~<ExpenseItem~~  ~~title={props.items[0].title}~~  ~~amount={props.items[0].amount}~~  ~~date={props.items[0].date}~~  ~~/>~~  ~~<ExpenseItem~~  ~~title={props.items[1].title}~~  ~~amount={props.items[1].amount}~~  ~~date={props.items[1].date}~~  ~~/>~~  ~~<ExpenseItem~~  ~~title={props.items[2].title}~~  ~~amount={props.items[2].amount}~~  ~~date={props.items[2].date}~~  ~~/>~~  ~~<ExpenseItem~~  ~~title={props.items[3].title}~~  ~~amount={props.items[3].amount}~~  ~~date={props.items[3].date}~~  ~~/>~~  </Card>  </div>  );  };  export default Expenses; |

|  |
| --- |
| src/components/Expenses/Expenses.js |
| import React, { useState } from "react";  import ExpenseItem from "./ExpenseItem";  import ExpensesFilter from "./ExpensesFilter";  import Card from "../UI/Card";  import "./Expenses.css";  const Expenses = (props) => {  const [filteredYear, setFilteredYear] = useState("2020");  const filterChangeHandler = (selectedYear) => {  setFilteredYear(selectedYear);  };  return (  <div>  <Card className="expenses">  <ExpensesFilter  selected={filteredYear}  onChangeFilter={filterChangeHandler}  />  {props.items.map((expense) => (  <ExpenseItem  title={expense.title}  amount={expense.amount}  date={expense.date}  />  ))}  </Card>  </div>  );  };  export default Expenses; |

If we now save this and reload the webpage, we get a warning, which we can ignore for now. We also see that we get all these items being rendered just as before but now it's actually happening dynamically, which simply means that now it's based on the actual array. This means that we can now also change the array and such changes will be reflected in that list, which is going to be what's talked about in the next lecture.



### 65. Using Stateful Lists

How can we now update this expenses array whenever a new expense is being added? For this we need to go to App.js because that is where we have the expenses array as well as the function addExpenseHandler, which is triggered whenever a new expense is added.

What we have to do is add this expense to

|  |
| --- |
| src/App.js |
| import React from "react";  import NewExpense from "./components/NewExpense/NewExpense";  import Expenses from "./components/Expenses/Expenses";  const App = () => {  const expenses = [  {  id: "e1",  title: "Toilet Paper",  amount: 94.12,  date: new Date(2020, 7, 14),  },  { id: "e2", title: "New TV", amount: 799.49, date: new Date(2021, 2, 12) },  {  id: "e3",  title: "Car Insurance",  amount: 294.67,  date: new Date(2021, 2, 28),  },  {  id: "e4",  title: "New Desk (Wooden)",  amount: 450,  date: new Date(2021, 5, 12),  },  ];  const addExpenseHandler = **expense** => {  console.log('In App.js');  console.log(expense);  }  return (  <div>  <NewExpense onAddExpense={addExpenseHandler}/>  <Expenses items={expenses} />  </div>  );  };  export default App; |

this expenses array. However, if we just add it, React won't update the component. Instead, to get React to update the component, you need to use state.

|  |
| --- |
| src/App.js |
| import React from "react";  import NewExpense from "./components/NewExpense/NewExpense";  import Expenses from "./components/Expenses/Expenses";  const App = () => {  **const expenses = [**  **{**  **id: "e1",**  **title: "Toilet Paper",**  **amount: 94.12,**  **date: new Date(2020, 7, 14),**  **},**  **{ id: "e2", title: "New TV", amount: 799.49, date: new Date(2021, 2, 12) },**  **{**  **id: "e3",**  **title: "Car Insurance",**  **amount: 294.67,**  **date: new Date(2021, 2, 28),**  **},**  **{**  **id: "e4",**  **title: "New Desk (Wooden)",**  **amount: 450,**  **date: new Date(2021, 5, 12),**  **},**  **];**  const addExpenseHandler = expense => {  console.log('In App.js');  console.log(expense);  }  return (  <div>  <NewExpense onAddExpense={addExpenseHandler}/>  <Expenses items={expenses} />  </div>  );  };  export default App; |

Now in App.js we want to import useState.

|  |
| --- |
| src/App.js |
| import React, **{useState}** from "react";  import NewExpense from "./components/NewExpense/NewExpense";  import Expenses from "./components/Expenses/Expenses";  const App = () => {  const expenses = [  {  id: "e1",  title: "Toilet Paper",  amount: 94.12,  date: new Date(2020, 7, 14),  },  { id: "e2", title: "New TV", amount: 799.49, date: new Date(2021, 2, 12) },  {  id: "e3",  title: "Car Insurance",  amount: 294.67,  date: new Date(2021, 2, 28),  },  {  id: "e4",  title: "New Desk (Wooden)",  amount: 450,  date: new Date(2021, 5, 12),  },  ];  const addExpenseHandler = expense => {  console.log('In App.js');  console.log(expense);  }  return (  <div>  <NewExpense onAddExpense={addExpenseHandler}/>  <Expenses items={expenses} />  </div>  );  };  export default App; |

We will actually grab the expenses array, extract it out of the App component function, store it above the App component function declaration, and name it DUMMY\_EXPENSES, as this is just some dummy data that we can use to initialize our state with.

|  |
| --- |
| src/App.js |
| import React, {useState} from "react";  import NewExpense from "./components/NewExpense/NewExpense";  import Expenses from "./components/Expenses/Expenses";  **const DUMMY\_EXPENSES = [**  **{**  **id: "e1",**  **title: "Toilet Paper",**  **amount: 94.12,**  **date: new Date(2020, 7, 14),**  **},**  **{ id: "e2", title: "New TV", amount: 799.49, date: new Date(2021, 2, 12) },**  **{**  **id: "e3",**  **title: "Car Insurance",**  **amount: 294.67,**  **date: new Date(2021, 2, 28),**  **},**  **{**  **id: "e4",**  **title: "New Desk (Wooden)",**  **amount: 450,**  **date: new Date(2021, 5, 12),**  **},**  **];**  const App = () => {  const addExpenseHandler = expense => {  console.log('In App.js');  console.log(expense);  }  return (  <div>  <NewExpense onAddExpense={addExpenseHandler}/>  <Expenses items={expenses} />  </div>  );  };  export default App; |

Inside of the App component function, we call useState, and I'll pass DUMMY\_EXPENSES as an initial state value, so that we have some initial expenses to display. We can then use destructuring to get access to the expenses and to the setExpenses state updating function.

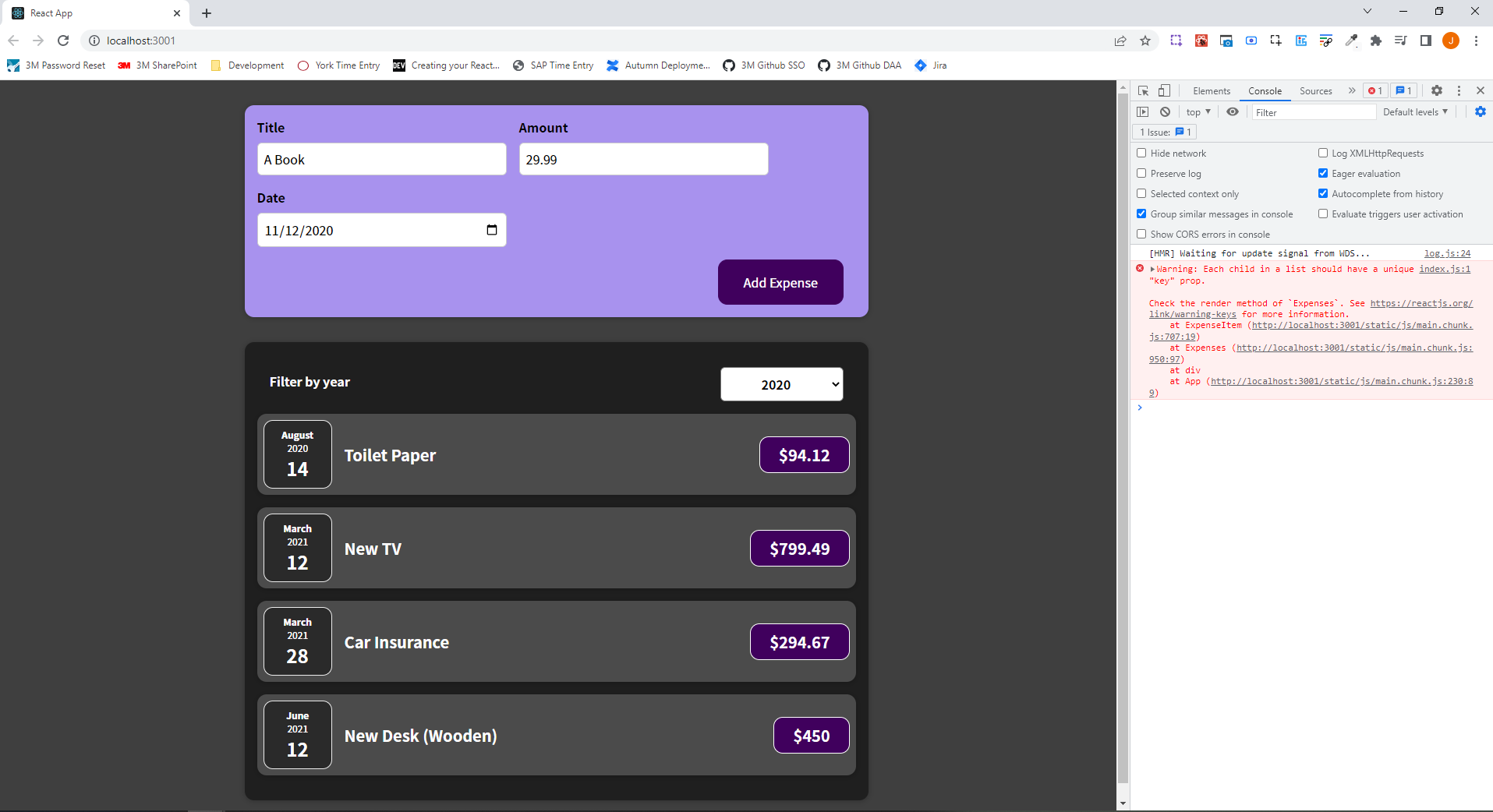
|  |
| --- |
| src/App.js |
| import React, {useState} from "react";  import NewExpense from "./components/NewExpense/NewExpense";  import Expenses from "./components/Expenses/Expenses";  const DUMMY\_EXPENSES = [  {  id: "e1",  title: "Toilet Paper",  amount: 94.12,  date: new Date(2020, 7, 14),  },  { id: "e2", title: "New TV", amount: 799.49, date: new Date(2021, 2, 12) },  {  id: "e3",  title: "Car Insurance",  amount: 294.67,  date: new Date(2021, 2, 28),  },  {  id: "e4",  title: "New Desk (Wooden)",  amount: 450,  date: new Date(2021, 5, 12),  },  ];  const App = () => {  **const [expenses, setExpenses] = useState(DUMMY\_EXPENSES);**  const addExpenseHandler = expense => {  console.log('In App.js');  console.log(expense);  }  return (  <div>  <NewExpense onAddExpense={addExpenseHandler}/>  <Expenses items={expenses} />  </div>  );  };  export default App; |

In the function addExpenseHandler, we can now remove the console.log() statements. In the function addExpenseHandler, where we want to add a new expense, we can then call setExpenses to set our expenses array to a new array which includes the new expense. Inside of setExpenses, we will update the state based on the previous state, or the previous snapshot of this state. As we learned, we should use the special function form for this state updating function, so it would be passed a function as an argument to this state updating function, and that function will automatically receive the latest state snapshot. So here we would then get our prevExpenses automatically by react and we would return our new array where we add this expense, which we're getting as a parameter here, and where we then add our prevExpenses with the spread operator.

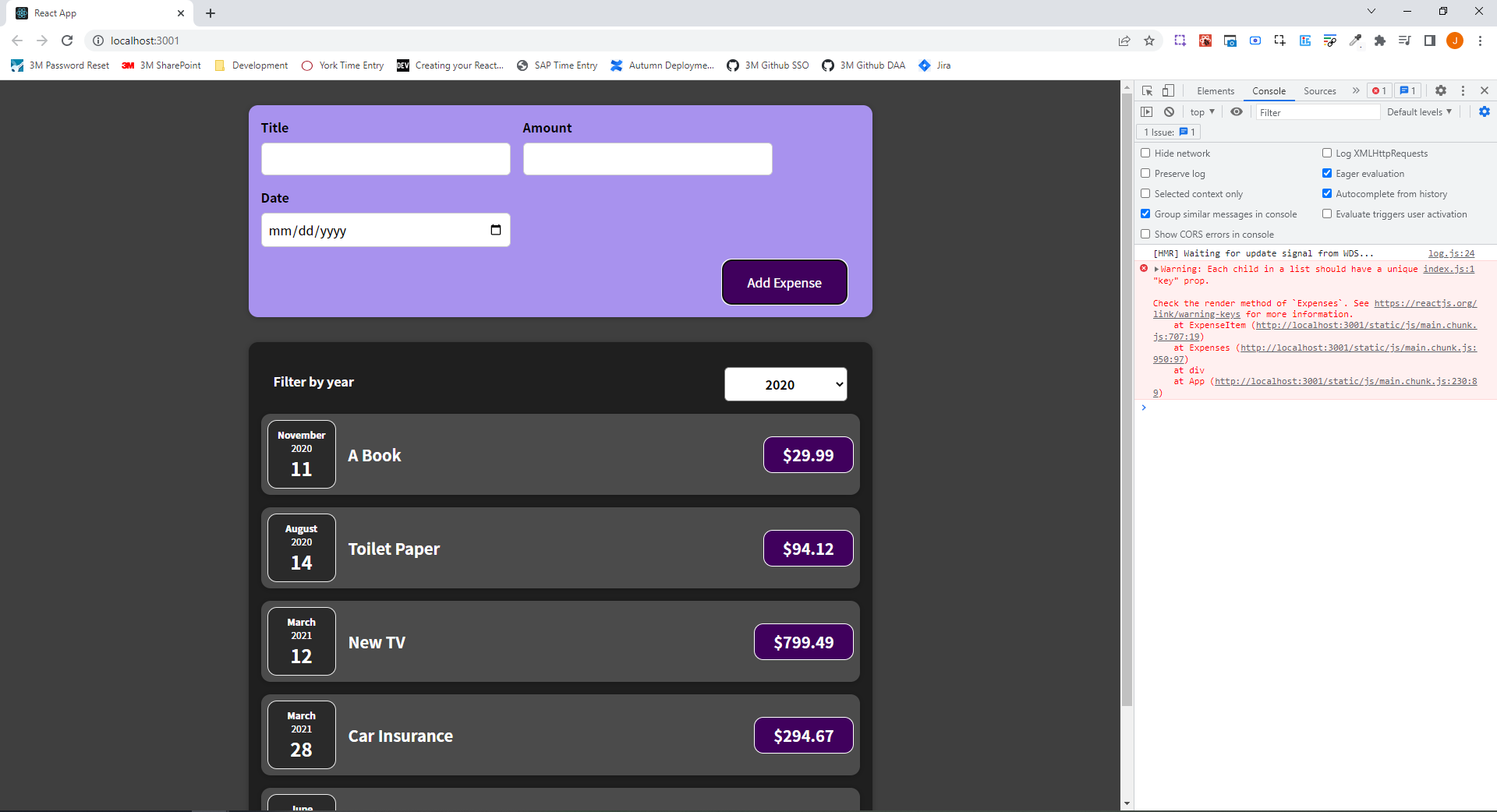
|  |
| --- |
| src/App.js |
| import React, {useState} from "react";  import NewExpense from "./components/NewExpense/NewExpense";  import Expenses from "./components/Expenses/Expenses";  const DUMMY\_EXPENSES = [  {  id: "e1",  title: "Toilet Paper",  amount: 94.12,  date: new Date(2020, 7, 14),  },  { id: "e2", title: "New TV", amount: 799.49, date: new Date(2021, 2, 12) },  {  id: "e3",  title: "Car Insurance",  amount: 294.67,  date: new Date(2021, 2, 28),  },  {  id: "e4",  title: "New Desk (Wooden)",  amount: 450,  date: new Date(2021, 5, 12),  },  ];  const App = () => {  const [expenses, setExpenses] = useState(DUMMY\_EXPENSES);  const addExpenseHandler = expense => {  **setExpenses((prevExpenses) => {**  **return [expense, ...prevExpenses];**  **});**  }  return (  <div>  <NewExpense onAddExpense={addExpenseHandler}/>  <Expenses items={expenses } />  </div>  );  };  export default App; |

We should now have a dynamic list, which automatically updates when items are added.

So now in the browser, if we refresh the page and we now add a book for $29.99, pick some date, and click "Add Expense",

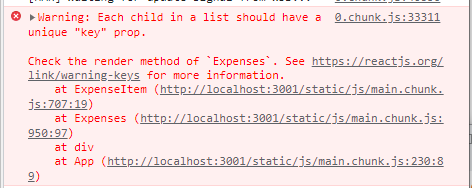


We will see A Book added as the first item. So now this is working and is truly a dynamic list by using state and by outputting the items using the map()method.



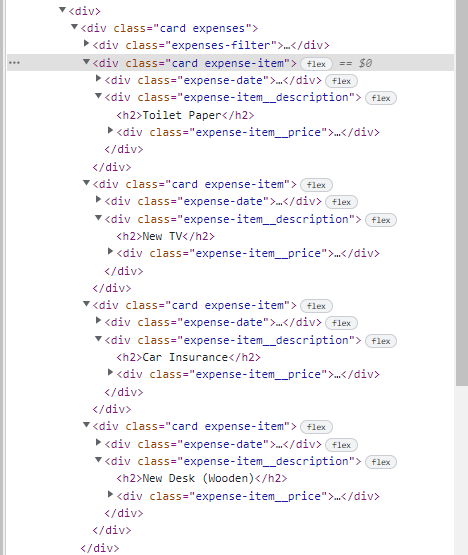
### 66. Understanding "Keys"

Why are we getting this "Key" warning? It looks like everything works.



Well React has a special concept when it comes to rendering lists of data, a concept which exists to ensure that React is able to update and render such lists efficiently without performance losses, or bugs, which may occur.

If we select the Elements tab in the Developer Tools, and hover over the list of items, we will see multiple divs representing each ExpenseItem.

****

If we add a new item, React renders this new item as the last item in the list of div's and updates all items and replaces their content, such that it again matches the order of the items in my Array. This is not great. This is happening because to React all of these items look similar, and it only sees that my Array changed; that my array is now longer than before. And, hence, it simply renders an additional div, and it adds that at the end. It then walks through all of the items and updates the content inside of every item to match the Array content again. The final result is correct but from a performance perspective, this is not great because all items are visited and updated, and it can even lead to bugs.

If the ExpenseItems would be stateful items and we would have some state managed inside of that, then if, for example, our first item, if it has a certain state, if we add a new item, the old first item would be overwritten with the new first item. Hence, any state changes we might’ve had in there would be lost. So besides potential performance issues, we could also run into bugs.

We have a way of telling React where a new item should be added. We do that by going to the place where we output our list of items,

|  |
| --- |
| src/components/Expenses/Expenses.js |
| import React, { useState } from "react";  import ExpenseItem from "./ExpenseItem";  import ExpensesFilter from "./ExpensesFilter";  import Card from "../UI/Card";  import "./Expenses.css";  const Expenses = (props) => {  const [filteredYear, setFilteredYear] = useState("2020");  const filterChangeHandler = (selectedYear) => {  setFilteredYear(selectedYear);  };  return (  <div>  <Card className="expenses">  <ExpensesFilter  selected={filteredYear}  onChangeFilter={filterChangeHandler}  />  **{props.items.map((expense) => (**  **<ExpenseItem**  **title={expense.title}**  **amount={expense.amount}**  **date={expense.date}**  **/>**  **))}**  </Card>  </div>  );  };  export default Expenses; |

We then add a special prop to this ExpenseItem here, and that's the key prop. The key prop is not a prop that we are using inside of ExpenseItem, but instead it's a prop you can add to any component no matter if it's a custom component by you or it's a built-in HTML element, you can always add this. And if you add the key prop, then you can help React identify the individual items. To identify the individual items, you need to set a unique value per list item, and that for example in this case here, would be expense.id because in the array called DUMMY\_EXPENSES holding our initial expenses, every item has a unique ID.

|  |
| --- |
| src/components/Expenses/Expenses.js |
| import React, { useState } from "react";  import ExpenseItem from "./ExpenseItem";  import ExpensesFilter from "./ExpensesFilter";  import Card from "../UI/Card";  import "./Expenses.css";  const Expenses = (props) => {  const [filteredYear, setFilteredYear] = useState("2020");  const filterChangeHandler = (selectedYear) => {  setFilteredYear(selectedYear);  };  return (  <div>  <Card className="expenses">  <ExpensesFilter  selected={filteredYear}  onChangeFilter={filterChangeHandler}  />  {props.items.map((expense) => (  <ExpenseItem  **key={expense.id}**  title={expense.title}  amount={expense.amount}  date={expense.date}  />  ))}  </Card>  </div>  );  };  export default Expenses; |

If you did not have a unique ID, you could use the second argument which you get here for the function you pass to map(), which is an automatically managed index. Though, it is discouraged to use that since with that you can still run into bugs because the index for a given item is always the same and not directly attached to the content of the item. For the ID that's different, as every item with certain content has a clear unique ID.

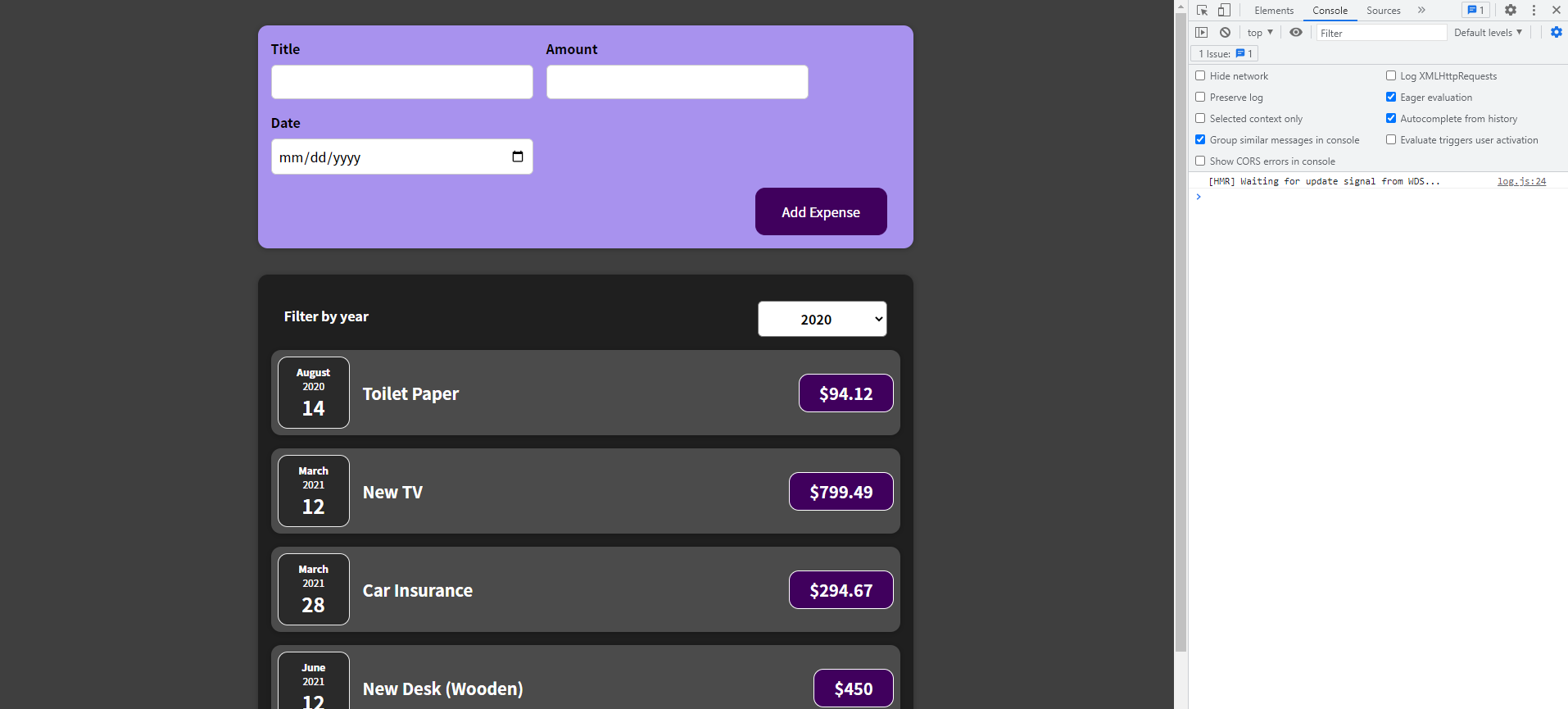
|  |
| --- |
| src/components/Expenses/Expenses.js |
| import React, { useState } from "react";  import ExpenseItem from "./ExpenseItem";  import ExpensesFilter from "./ExpensesFilter";  import Card from "../UI/Card";  import "./Expenses.css";  const Expenses = (props) => {  const [filteredYear, setFilteredYear] = useState("2020");  const filterChangeHandler = (selectedYear) => {  setFilteredYear(selectedYear);  };  return (  <div>  <Card className="expenses">  <ExpensesFilter  selected={filteredYear}  onChangeFilter={filterChangeHandler}  />  {props.items.map((expense, **index**) => (  <ExpenseItem  key={**index**}  title={expense.title}  amount={expense.amount}  date={expense.date}  />  ))}  </Card>  </div>  );  };  export default Expenses; |

In reality, it turns out that in most scenarios you do have some unique ID because typically you are rendering some data which comes from a database or anything like that. And there you work with unique IDs anyway. So finding some unique identifier is no problem, and you can use any primitive value as a unique identifier.

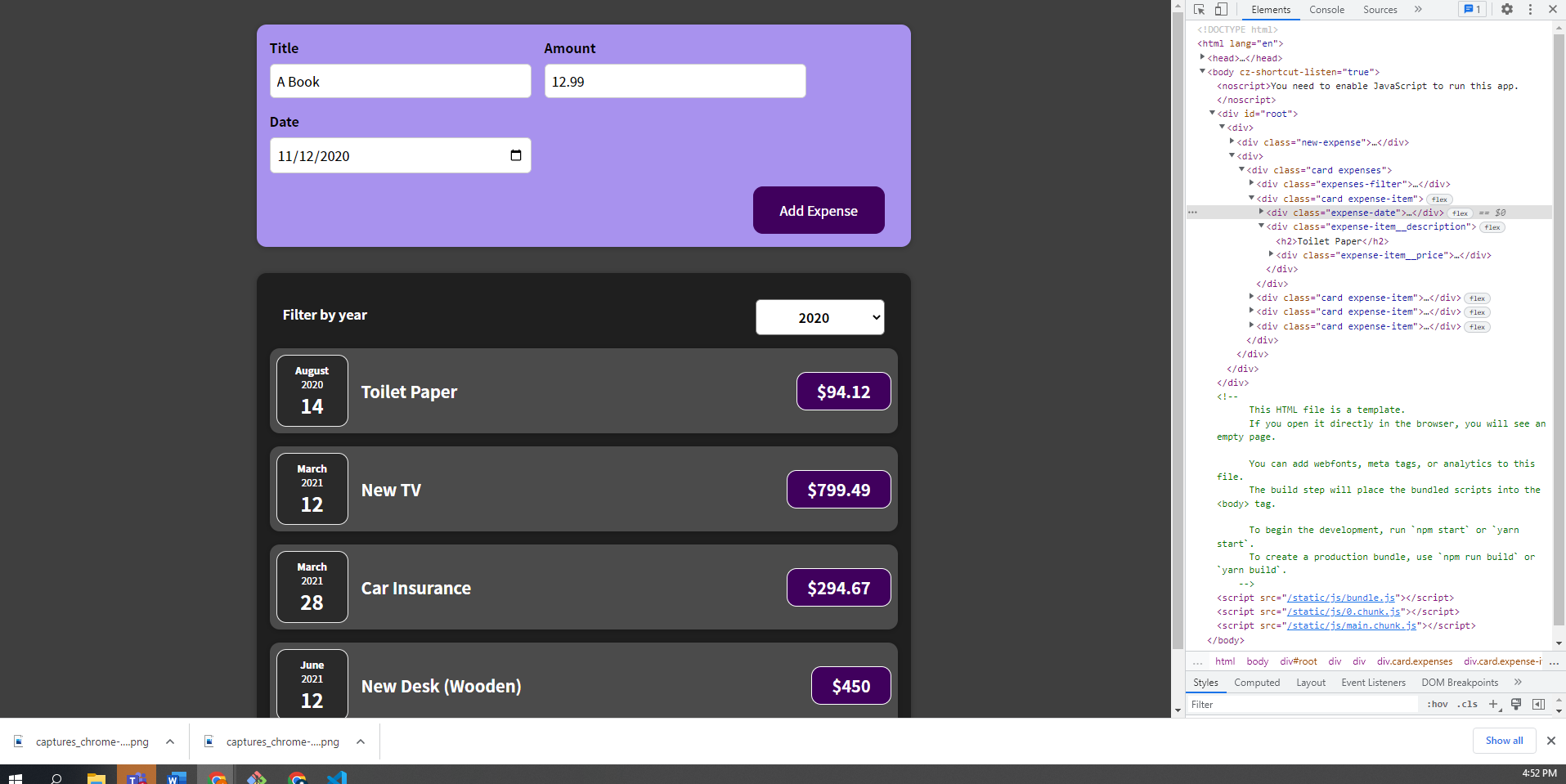
Now, with the key prop pointing at expense.id,

|  |
| --- |
| src/components/Expenses/Expenses.js |
| import React, { useState } from "react";  import ExpenseItem from "./ExpenseItem";  import ExpensesFilter from "./ExpensesFilter";  import Card from "../UI/Card";  import "./Expenses.css";  const Expenses = (props) => {  const [filteredYear, setFilteredYear] = useState("2020");  const filterChangeHandler = (selectedYear) => {  setFilteredYear(selectedYear);  };  return (  <div>  <Card className="expenses">  <ExpensesFilter  selected={filteredYear}  onChangeFilter={filterChangeHandler}  />  {props.items.map((expense) => (  <ExpenseItem  **key={expense.id}**  title={expense.title}  amount={expense.amount}  date={expense.date}  />  ))}  </Card>  </div>  );  };  export default Expenses; |

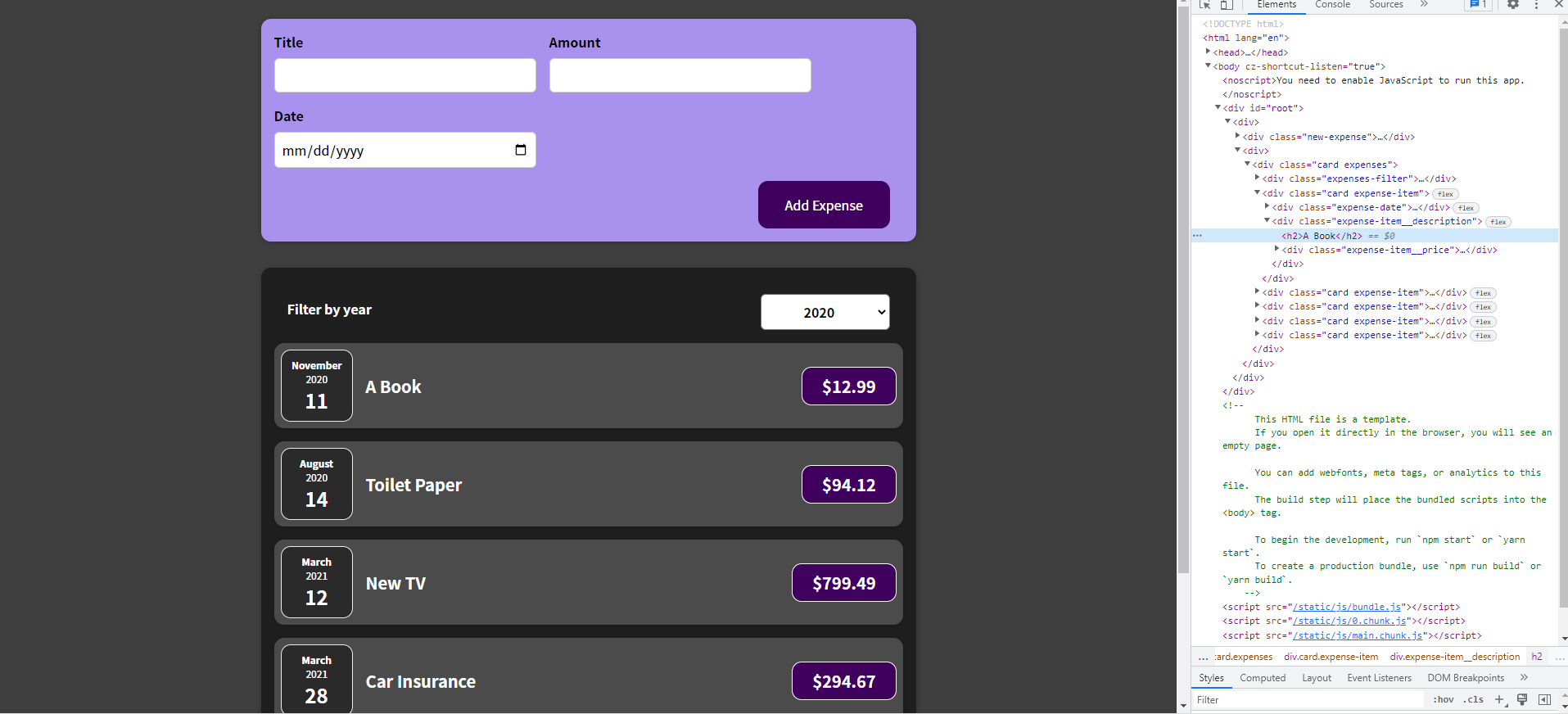
if we reload the page, we no longer get that warning.



If we inspect the first item in the list and add a new expense,



We see the div with the item just added at the beginning of the list of divs correctly, as it should be, because now React is able to uniquely identify all these items, and it is therefore aware, not just how long the array is, but also which items should be placed. It's also able to update this list in a more efficient manner.



You should always add a key prop when mapping out lists of items.

### Assignment 3: Time to Practice: Working with Lists

I need to make the year filter work, so that if a user selects a year value, the list is filtered correctly. So, if I selected an expense from 2021, I only see expenses showing up from 2021 in the rendered list of expenses.

Two hints while working on this:

1. You can filter arrays with the filter method. Like map, filter is a built-in method for arrays in JavaScript.
2. You should not think to complicated here. You should not change the overall expenses array. Instead, you may want to just derive a new array based on the full expenses array. The new array should be a subset of the expenses array for the chosen filter.

### 67. Outputting Conditional Content

Our filter is working but we can still select values from the dropdown where we don't have an expense for the particular year selected. We might want to show a message in such cases where we select years where we don't have expenses for the particular year selected.

Conditional content is about rendering different output under different conditions. In Expenses.js, we render our list of ExpenseItems, but if our filteredExpenses array turns out to be empty, we render nothing. We might want to render a message telling the user that we have no items for the chosen filter. For that, we want to render content conditionally.

We can add a dynamic expression in our JSX code and simply use a condition. We are not talking about an if condition or for loops are not allowed between curly braces because they are too long, but we can use a ternary expression. We can check if filteredExpenses.length is equal to 0 meaning that we have no items in that filteredExpenses array. If that's the case, after the question mark, which is the default JavaScript ternary expression syntax, we render a simple message in a paragraph where we say "No expenses found." Otherwise, after the colon, we render our list.

|  |
| --- |
| src/components/Expenses/Expenses.js |
| import React, { useState } from "react";  import ExpenseItem from "./ExpenseItem";  import ExpensesFilter from "./ExpensesFilter";  import Card from "../UI/Card";  import "./Expenses.css";  const Expenses = (props) => {  const [filteredYear, setFilteredYear] = useState("2020");  const filterChangeHandler = (selectedYear) => {  setFilteredYear(selectedYear);  };  const filteredExpenses = props.items.filter((expense) => {  return expense.date.getFullYear().toString() === filteredYear;  });  return (  <div>  <Card className="expenses">  <ExpensesFilter  selected={filteredYear}  onChangeFilter={filterChangeHandler}  />  **{filteredExpenses.length === 0 ? (**  **<p>No expenses Found</p>**  **) : (**  **filteredExpenses.map((expense) => (**  **<ExpenseItem**  **key={expense.id}**  **title={expense.title}**  **amount={expense.amount}**  **date={expense.date}**  **/>**  **))**  **)}**  </Card>  </div>  );  };  export default Expenses; |

Long ternary expressions like this can be a bit hard to read, so we can simplify this by doing the following:

{filteredExpenses.length === 0 && <p>No expenses found.</p>

We can add the and operator and then use the JSX content we want to render if the length of filteredExpenses is equal to 0. In JavaScript, if you use the && (and) operator, it will return the part after the && operator as a result of the overall check if the first condition (if the length of filteredExpenses is equal to 0) is met. It moves onto the part after the and operator and it then returns that value.

We can now do this same thing when filteredExpenses is greater than 0.

|  |
| --- |
| src/components/Expenses/Expenses.js |
| import React, { useState } from "react";  import ExpenseItem from "./ExpenseItem";  import ExpensesFilter from "./ExpensesFilter";  import Card from "../UI/Card";  import "./Expenses.css";  const Expenses = (props) => {  const [filteredYear, setFilteredYear] = useState("2020");  const filterChangeHandler = (selectedYear) => {  setFilteredYear(selectedYear);  };  const filteredExpenses = props.items.filter((expense) => {  return expense.date.getFullYear().toString() === filteredYear;  });  return (  <div>  <Card className="expenses">  <ExpensesFilter  selected={filteredYear}  onChangeFilter={filterChangeHandler}  />  {filteredExpenses.length === 0 && <p>No expenses found.</p>}  **{filteredExpenses.length > 0 &&**  **filteredExpenses.map((expense) => (**  **<ExpenseItem**  **key={expense.id}**  **title={expense.title}**  **amount={expense.amount}**  **date={expense.date}**  **/>**  **))}**  </Card>  </div>  );  };  export default Expenses; |

This might even be too much logic in the JSX code, so we have an alternative way of handling this. We can create a variable with a default value like so:

|  |
| --- |
| src/components/Expenses/Expenses.js |
| import React, { useState } from "react";  import ExpenseItem from "./ExpenseItem";  import ExpensesFilter from "./ExpensesFilter";  import Card from "../UI/Card";  import "./Expenses.css";  const Expenses = (props) => {  const [filteredYear, setFilteredYear] = useState("2020");  const filterChangeHandler = (selectedYear) => {  setFilteredYear(selectedYear);  };  const filteredExpenses = props.items.filter((expense) => {  return expense.date.getFullYear().toString() === filteredYear;  });  **let expensesContent = <p>No expenses found.</p>**  return (  <div>  <Card className="expenses">  <ExpensesFilter  selected={filteredYear}  onChangeFilter={filterChangeHandler}  />  {filteredExpenses.length === 0 && <p>No expenses found.</p>}  {filteredExpenses.length > 0 &&  filteredExpenses.map((expense) => (  <ExpenseItem  key={expense.id}  title={expense.title}  amount={expense.amount}  date={expense.date}  />  ))}  </Card>  </div>  );  };  export default Expenses; |

We can store JSX content in variables. We store it in a variable here specifically because now we can add an if check before we return. We can check if filteredExpenses.length is greater than 0. If filteredExpenses is greater than 0, we can set expensesContent to the value of the map call.

|  |
| --- |
| src/components/Expenses/Expenses.js |
| import React, { useState } from "react";  import ExpenseItem from "./ExpenseItem";  import ExpensesFilter from "./ExpensesFilter";  import Card from "../UI/Card";  import "./Expenses.css";  const Expenses = (props) => {  const [filteredYear, setFilteredYear] = useState("2020");  const filterChangeHandler = (selectedYear) => {  setFilteredYear(selectedYear);  };  const filteredExpenses = props.items.filter((expense) => {  return expense.date.getFullYear().toString() === filteredYear;  });  let expensesContent = <p>No expenses found.</p>  **if (filteredExpenses.length > 0){**  **expensesContent = filteredExpenses.map((expense) => (**  **<ExpenseItem**  **key={expense.id}**  **title={expense.title}**  **amount={expense.amount}**  **date={expense.date}**  **/>**  **))**  **}**  return (  <div>  <Card className="expenses">  <ExpensesFilter  selected={filteredYear}  onChangeFilter={filterChangeHandler}  />  {filteredExpenses.length === 0 && <p>No expenses found.</p>}  {filteredExpenses.length > 0 &&  filteredExpenses.map((expense) => (  <ExpenseItem  key={expense.id}  title={expense.title}  amount={expense.amount}  date={expense.date}  />  ))}  </Card>  </div>  );  };  export default Expenses; |

In our returned JSX code, we can get rid of all that logic, and we simply add a dynamic expression where we point at expensesContent.

|  |
| --- |
| src/components/Expenses/Expenses.js |
| import React, { useState } from "react";  import ExpenseItem from "./ExpenseItem";  import ExpensesFilter from "./ExpensesFilter";  import Card from "../UI/Card";  import "./Expenses.css";  const Expenses = (props) => {  const [filteredYear, setFilteredYear] = useState("2020");  const filterChangeHandler = (selectedYear) => {  setFilteredYear(selectedYear);  };  const filteredExpenses = props.items.filter((expense) => {  return expense.date.getFullYear().toString() === filteredYear;  });  let expensesContent = <p>No expenses found.</p>  if (filteredExpenses.length > 0){  expensesContent = filteredExpenses.map((expense) => (  <ExpenseItem  key={expense.id}  title={expense.title}  amount={expense.amount}  date={expense.date}  />  ))  }  return (  <div>  <Card className="expenses">  <ExpensesFilter  selected={filteredYear}  onChangeFilter={filterChangeHandler}  />  **{expensesContent}**  </Card>  </div>  );  };  export default Expenses; |

### 68. Adding Conditional Return Statements

We are going to extract the list logic into a new component to make the Expenses component a little leaner. We will create an "ExpensesList.js" file and an "ExpensesList.css" file. We will copy the css code given into "ExpenseList.css" and then in "ExpensesList.js" we will write the following code:

|  |
| --- |
| src/components/Expenses/ExpensesList.js |
| import React from 'react';  const ExpensesList = props => {  }  export default ExpensesList; |

I will take the fallback content and the if check and move that into the ExpensesList component:

|  |
| --- |
| src/components/Expenses/ExpensesList.js |
| import React from 'react';  import ExpenseItem from './ExpenseItem';  const ExpensesList = props => {  let expensesContent = <p>No expenses found.</p>  if (filteredExpenses.length > 0){  expensesContent = filteredExpenses.map((expense) => (  <ExpenseItem  key={expense.id}  title={expense.title}  amount={expense.amount}  date={expense.date}  />  ))  }  }  export default ExpensesList; |

The variable filteredExpenses no longer exists here. We just use props.items instead.

|  |
| --- |
| src/components/Expenses/ExpensesList.js |
| import React from 'react';  import ExpenseItem from './ExpenseItem';  const ExpensesList = props => {  let expensesContent = <p>No expenses found.</p>  if (**props.items**.length > 0){  expensesContent = **props.items**.map((expense) => (  <ExpenseItem  key={expense.id}  title={expense.title}  amount={expense.amount}  date={expense.date}  />  ))  }  }  export default ExpensesList; |

This means that in Expenses.js we now have the following:

|  |
| --- |
| src/components/Expenses/Expenses.js |
| import React, { useState } from "react";  import ExpensesFilter from "./ExpensesFilter";  import ExpensesList from "./ExpensesList";  import Card from "../UI/Card";  import "./Expenses.css";  const Expenses = (props) => {  const [filteredYear, setFilteredYear] = useState("2020");  const filterChangeHandler = (selectedYear) => {  setFilteredYear(selectedYear);  };  const filteredExpenses = props.items.filter((expense) => {  return expense.date.getFullYear().toString() === filteredYear;  });  return (  <div>  <Card className="expenses">  <ExpensesFilter  selected={filteredYear}  onChangeFilter={filterChangeHandler}  />  **<ExpensesList items={filteredExpenses}/>**  </Card>  </div>  );  };  export default Expenses; |

In "ExpensesList.js" we want to make some changes. We want to return some JSX, and we will return an unordered list, giving it a className of "expenses-list".

|  |
| --- |
| src/components/Expenses/ExpensesList.js |
| import React from 'react';  import ExpenseItem from './ExpenseItem';  import './ExpensesList.css';  const ExpensesList = props => {  let expensesContent = <p>No expenses found.</p>  if (props.items.length > 0){  expensesContent = props.items.map((expense) => (  <ExpenseItem  key={expense.id}  title={expense.title}  amount={expense.amount}  date={expense.date}  />  ))  }  **return <ul className="expensesList">**  **</ul>**  }  export default ExpensesList; |

Inside of the unordered list, I want to output my map here, my ExpenseItem array.

|  |
| --- |
| src/components/Expenses/ExpensesList.js |
| import React from 'react';  import ExpenseItem from './ExpenseItem';  import './ExpensesList.css';  const ExpensesList = props => {  let expensesContent = <p>No expenses found.</p>  if (props.items.length > 0){  expensesContent = props.items.map((expense) => (  <ExpenseItem  key={expense.id}  title={expense.title}  amount={expense.amount}  date={expense.date}  />  ))  }  return <ul className="expensesList">  **{**  **props.items.map((expense) => (**  **<ExpenseItem**  **key={expense.id}**  **title={expense.title}**  **amount={expense.amount}**  **date={expense.date}**  **/>**  **))**  **}**  </ul>  }  export default ExpensesList; |

We will now handle the if check a little differently. We will add the if check, but we will check if props.items.length equals 0, so if we have no items, and if that is the case we will return a totally different JSX snippet because that's yet another way of handling conditional content. If what your component returns changes entirely based on different conditions, you can use this approach. It would not have been appropriate in "Expenses.js" because only a part of the JSX snippet which we returned changed. If your entire JSX content changes when data is missing, you can also add an if check where you return a different JSX block if some condition is met. So here, we can return an h2 element with a className of "expenses-list\_\_fallback" with a text of "Found no expenses." We can also get rid of the variable called expensesContent.

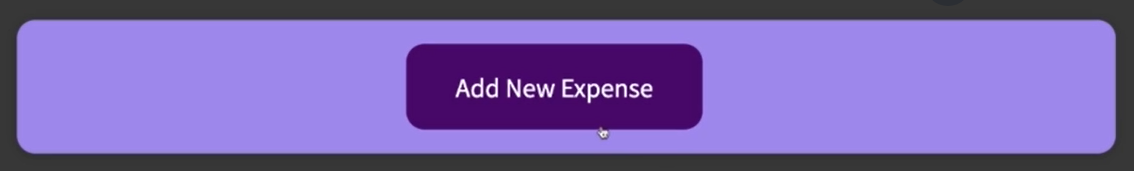
|  |
| --- |
| src/components/Expenses/ExpensesList.js |
| import React from 'react';  import ExpenseItem from './ExpenseItem';  import './ExpensesList.css';  const ExpensesList = props => {  **if (props.items.length === 0) {**  **return <h2 className="expenses-list\_\_fallback">Found no expenses.</h2>**  **}**  return <ul className="expensesList">  {  props.items.map((expense) => (  <ExpenseItem  key={expense.id}  title={expense.title}  amount={expense.amount}  date={expense.date}  />  ))  }  </ul>  }  export default ExpensesList; |

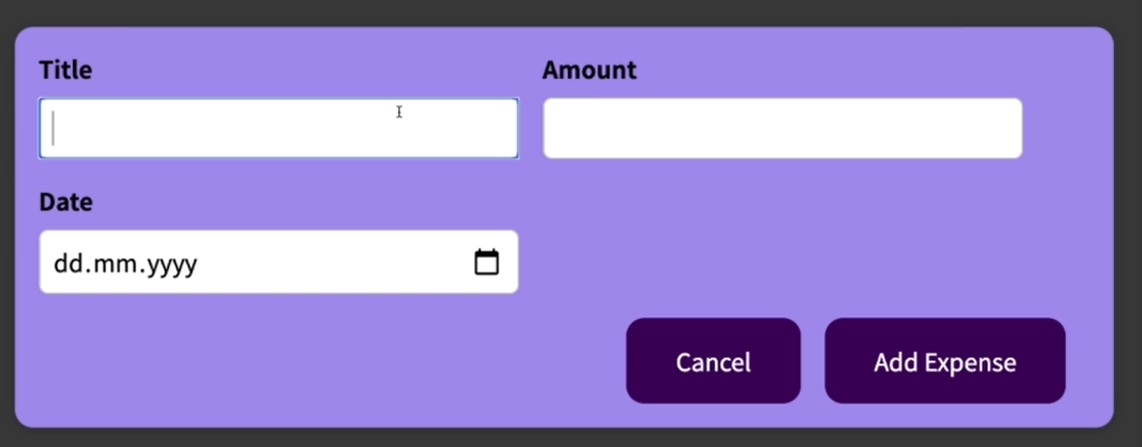
Since we are using an unordered list in the ExpenseList component, for semantic reasons, in the ExpenseItem component, we will switch away from a div being rendered to a list item instead.

|  |
| --- |
| src/components/Expenses/ExpenseItem.js |
| import React, { useState } from "react";  import ExpenseDate from "./ExpenseDate";  import Card from "../UI/Card";  import "./ExpenseItem.css";  const ExpenseItem = (props) => {  return (  **<li>**  <Card className="expense-item">  <ExpenseDate date={props.date} />  <div className="expense-item\_\_description">  <h2>{props.title}</h2>  <div className="expense-item\_\_price">${props.amount}</div>  </div>  </Card>  **</li>**  );  };  export default ExpenseItem; |

### Assignment 4: Time to Practice: Conditional Content

The form is always currently shown. We need to change the logic to make sure that we have a button that can be shown instead of the form where the button says something like "Add New Expense". When the button is clicked, the button disappears and the form is shown instead, and when the form's add expense button is clicked, the form disappears again, and the button which opens the form is shown again.



You should also add a cancel button to this form, which also makes the form disappear without submitting it. 

#### Plan for Assignment 4: Time to Practice: Conditional Content

~~Add the "Add Expense" button to a separate component called ExpenseButton. Add the ExpenseButton component, to the ExpenseForm component.~~

In the ExpenseForm component, create a state hook:

const [formVisable, setFormVisable] = useState(false);

Wrap JSX in the return with a JSX fragment.

Check if {if !formVisable{ <button onClick={showForm}>Add Expense</button>}}

In show showForm, call setFormVisable(true);

Create else condition to show the form.

Create a cancel button inside the form, which has an onClick event that points at the function hideForm. In the hideForm function, call setFormVisable(false);

In the submitHandler method, after event.preventDefault();, call setFormVisable(false);

Code I ended up writing:

|  |
| --- |
| src/components/NewExpense/ExpenseForm.js |
| import React, { useState } from "react";  import "./ExpenseForm.css";  const ExpenseForm = (props) => {  const [enteredTitle, setEnteredTitle] = useState("");  const [enteredAmount, setEnteredAmount] = useState("");  const [enteredDate, setEnteredDate] = useState("");  **const [formHidden, setFormHidden] = useState(true);**  const titleChangeHandler = (event) => {  setEnteredTitle(event.target.value);  };  const amountChangeHandler = (event) => {  const dateParts = event.target.value.split("-");  setEnteredAmount(dateParts[0], dateParts[1] + 1, dateParts[2]);  };  const dateChangeHandler = (event) => {  setEnteredDate(event.target.value);  };  // Handle date issue  const dateString = enteredDate;  const dateParts = dateString.split("-");  const aDate = new Date(dateParts[0], dateParts[1] - 1, dateParts[2]);  const submitHandler = (event) => {  event.preventDefault();  **setFormHidden(true);**  const expenseData = {  title: enteredTitle,  amount: enteredAmount,  date: aDate,  };  props.onSaveExpenseData(expenseData);  setEnteredTitle("");  setEnteredAmount("");  setEnteredDate("");  };  **const showForm = () => {**  **setFormHidden(false);**  **}**  **const hideForm = () => {**  **setFormHidden(true);**  **}**    **const expenseButton = <button onClick={showForm}>Add Expense</button>**  **const expenseForm =** <form onSubmit={submitHandler}>  <div className="new-expense\_\_controls">  <div className="new-expense\_\_control">  <label>Title</label>  <input  type="text"  value={enteredTitle}  onChange={titleChangeHandler}  />  </div>  <div className="new-expense\_\_control">  <label>Amount</label>  <input  type="number"  value={enteredAmount}  min="0.01"  step="0.01"  onChange={amountChangeHandler}  />  </div>  <div className="new-expense\_\_control">  <label>Date</label>  <input  type="date"  value={enteredDate}  min="2019-01-01"  max="2022-12-31"  onChange={dateChangeHandler}  />  </div>  </div>  <div className="new-expense\_\_actions">  **<button onClick={hideForm}>Cancel</button>**  <button type="submit">Add Expense</button>  </div>  </form>  return (  **<>**  **{ formHidden ? expenseButton : expenseForm }**  **</>**  );  };  export default ExpenseForm; |

#### Notes from Max's solution video

In NewExpense, we want to add <button>Add New Expense</button>. When the "Add New Expense" button is clicked, we want to show the form, and when the form is submitted or canceled, the button should be shown. That means that we need to register a new state here because we'll have a brand new state for our application or this part of the application.

|  |
| --- |
| src/components/NewExpense/NewExpense.js |
| import React from "react";  import ExpenseForm from "./ExpenseForm";  import "./NewExpense.css";  const NewExpense = (props) => {  const saveExpenseDataHandler = (enteredExpenseData) => {  const expenseData = {  ...enteredExpenseData,  id: Math.random().toString(),  };  props.onAddExpense(expenseData);  };  return (  <div className="new-expense">  **<button>Add New Expense</button>**  <ExpenseForm onSaveExpenseData={saveExpenseDataHandler} />  </div>  );  };  export default NewExpense; |

We have the state that we're currently adding or editing in expense. We have an editing state where the form should be open and we have another state where we are currently not working on an expense where the button should be shown instead. We'll have a brand new state which should be conditionally rendered on the screen, and, therefore, we need to add state to our component, a new state variable. Let's import useState from 'react';

|  |
| --- |
| src/components/NewExpense/NewExpense.js |
| import React, **{useState}** from "react";  import ExpenseForm from "./ExpenseForm";  import "./NewExpense.css";  const NewExpense = (props) => {  const saveExpenseDataHandler = (enteredExpenseData) => {  const expenseData = {  ...enteredExpenseData,  id: Math.random().toString(),  };  props.onAddExpense(expenseData);  };  return (  <div className="new-expense">  <button>Add New Expense</button>  <ExpenseForm onSaveExpenseData={saveExpenseDataHandler} />  </div>  );  };  export default NewExpense; |

We will call the state, and we basically just need a true or false state which states whether the form shall be shown or not. Initially, we could start with false, and we could name this state isEditing and setIsEditing, which is the state updating function.

|  |
| --- |
| src/components/NewExpense/NewExpense.js |
| import React, {useState} from "react";  import ExpenseForm from "./ExpenseForm";  import "./NewExpense.css";  const NewExpense = (props) => {  **const [isEditing, setIsEditing] = useState(false);**  const saveExpenseDataHandler = (enteredExpenseData) => {  const expenseData = {  ...enteredExpenseData,  id: Math.random().toString(),  };  props.onAddExpense(expenseData);  };  return (  <div className="new-expense">  <button>Add New Expense</button>  <ExpenseForm onSaveExpenseData={saveExpenseDataHandler} />  </div>  );  };  export default NewExpense; |

We then add a new function called startEditingHandler, which calls setIsEditing(true);, which sets isEditing to true.

|  |
| --- |
| src/components/NewExpense/NewExpense.js |
| import React, {useState} from "react";  import ExpenseForm from "./ExpenseForm";  import "./NewExpense.css";  const NewExpense = (props) => {  const [isEditing, setIsEditing] = useState(false);  const saveExpenseDataHandler = (enteredExpenseData) => {  const expenseData = {  ...enteredExpenseData,  id: Math.random().toString(),  };  props.onAddExpense(expenseData);  };  **const startEditingHandler = () => {**  **setIsEditing(true);**  **}**  return (  <div className="new-expense">  <button>Add New Expense</button>  <ExpenseForm onSaveExpenseData={saveExpenseDataHandler} />  </div>  );  };  export default NewExpense; |

The function startEditingHandler should be triggered when the "Add New Expense" button is clicked, which then in turn should lead to the ExpenseForm showing up. Therefore, on this button, we can add the onClick prop and point at the startEditingHandler function when it's clicked, so that isEditing is set to true when this button is clicked.

|  |
| --- |
| src/components/NewExpense/NewExpense.js |
| import React, {useState} from "react";  import ExpenseForm from "./ExpenseForm";  import "./NewExpense.css";  const NewExpense = (props) => {  const [isEditing, setIsEditing] = useState(false);  const saveExpenseDataHandler = (enteredExpenseData) => {  const expenseData = {  ...enteredExpenseData,  id: Math.random().toString(),  };  props.onAddExpense(expenseData);  };  const startEditingHandler = () => {  setIsEditing(true);  }  return (  <div className="new-expense">  <button **onClick={startEditingHandler}**>Add New Expense</button>  <ExpenseForm onSaveExpenseData={saveExpenseDataHandler} />  </div>  );  };  export default NewExpense; |

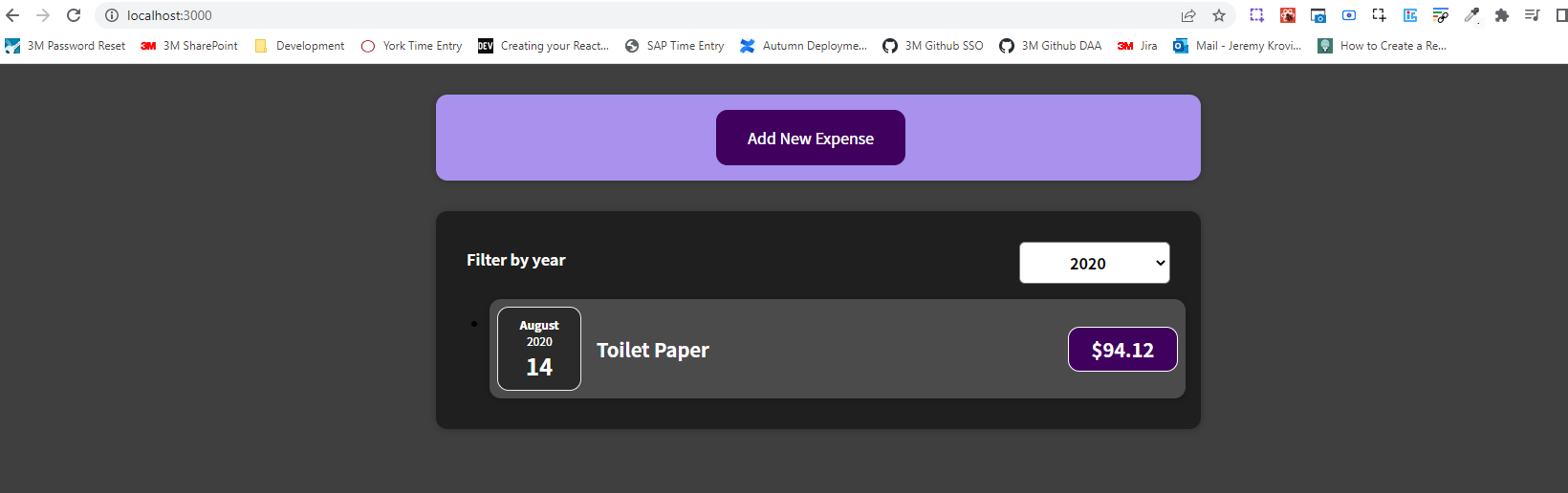
Now, we can use the isEditing state to control which of the two elements here—the button or the ExpenseForm—is shown. We want to show the button if we are not editing, hence the exclamation mark at the beginning, and we will do this in one line using &&.

|  |
| --- |
| src/components/NewExpense/NewExpense.js |
| import React, {useState} from "react";  import ExpenseForm from "./ExpenseForm";  import "./NewExpense.css";  const NewExpense = (props) => {  const [isEditing, setIsEditing] = useState(false);  const saveExpenseDataHandler = (enteredExpenseData) => {  const expenseData = {  ...enteredExpenseData,  id: Math.random().toString(),  };  props.onAddExpense(expenseData);  };  const startEditingHandler = () => {  setIsEditing(true);  }  return (  <div className="new-expense">  **{!isEditing &&** <button onClick={startEditingHandler}>Add New Expense</button>**}**  <ExpenseForm onSaveExpenseData={saveExpenseDataHandler} />  </div>  );  };  export default NewExpense; |

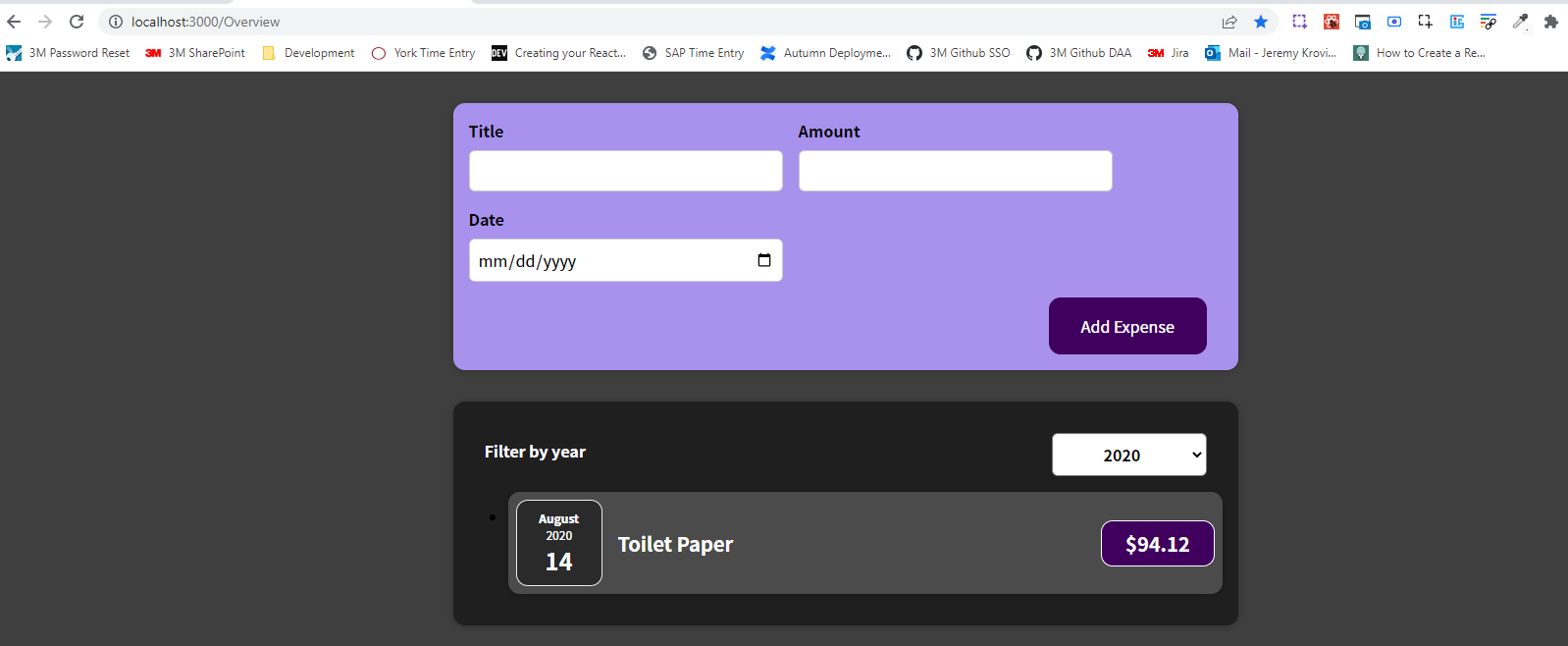
Of course you could also work with a variable that has a default value, or an if check, or with a ternary expression. Instead, we will have two dynamic expressions. Check if isEditing is false, hence we show the button, and check if isEditing is true, in which case we show the ExpenseForm component.

|  |
| --- |
| src/components/NewExpense/NewExpense.js |
| import React, {useState} from "react";  import ExpenseForm from "./ExpenseForm";  import "./NewExpense.css";  const NewExpense = (props) => {  const [isEditing, setIsEditing] = useState(false);  const saveExpenseDataHandler = (enteredExpenseData) => {  const expenseData = {  ...enteredExpenseData,  id: Math.random().toString(),  };  props.onAddExpense(expenseData);  };  const startEditingHandler = () => {  setIsEditing(true);  }  return (  <div className="new-expense">  {!isEditing && <button onClick={startEditingHandler}>Add New Expense</button>}  **{isEditing &&** <ExpenseForm onSaveExpenseData={saveExpenseDataHandler} />**}**  </div>  );  };  export default NewExpense; |

Now if we save and reload we will see just the button



If we click the button, we will see the form



Inside of the form, we now also want to add a "Cancel" button and make sure that when the form is submitted, we also stop editing. In ExpenseForm, we can add a button side-by-side with the "Add Expense" button and give it the text cancel, set the type to button, so that the button does not submit the form and add a onClick handler to execute a function when the "Cancel" button is pressed.

|  |
| --- |
| src/components/NewExpense/ExpenseForm.js |
| import React, { useState } from "react";  import "./ExpenseForm.css";  const ExpenseForm = (props) => {  const [enteredTitle, setEnteredTitle] = useState("");  const [enteredAmount, setEnteredAmount] = useState("");  const [enteredDate, setEnteredDate] = useState("");  // const [userInput, setUserInput] = useState({  // enteredTitle: '',  // enteredAmount: '',  // enteredDate: ''  // });  const titleChangeHandler = (event) => {  setEnteredTitle(event.target.value);  // setUserInput({  // ...userInput,  // enteredTitle: event.target.value,  // })  // setUserInput((prevState) => {  // return { ...prevState, enteredTitle: event.target.value}  // })  };  const amountChangeHandler = (event) => {  const dateParts = event.target.value.split("-");  setEnteredAmount(dateParts[0], dateParts[1] + 1, dateParts[2]);  // setUserInput({  // ...userInput,  // enteredAmount: event.target.value,  // })  };  const dateChangeHandler = (event) => {  setEnteredDate(event.target.value);  // setUserInput({  // ...userInput,  // enteredDate: event.target.value,  // })  };  // Handle date issue  const dateString = enteredDate;  const dateParts = dateString.split("-");  const aDate = new Date(dateParts[0], dateParts[1] - 1, dateParts[2]);  const submitHandler = (event) => {  event.preventDefault();  const expenseData = {  title: enteredTitle,  amount: enteredAmount,  date: aDate,  };  props.onSaveExpenseData(expenseData);  setEnteredTitle("");  setEnteredAmount("");  setEnteredDate("");  };  return (  <form onSubmit={submitHandler}>  <div className="new-expense\_\_controls">  <div className="new-expense\_\_control">  <label>Title</label>  <input  type="text"  value={enteredTitle}  onChange={titleChangeHandler}  />  </div>  <div className="new-expense\_\_control">  <label>Amount</label>  <input  type="number"  value={enteredAmount}  min="0.01"  step="0.01"  onChange={amountChangeHandler}  />  </div>  <div className="new-expense\_\_control">  <label>Date</label>  <input  type="date"  value={enteredDate}  min="2019-01-01"  max="2022-12-31"  onChange={dateChangeHandler}  />  </div>  </div>  <div className="new-expense\_\_actions">  **<button type="button" onClick={}>Cancel</button>**  <button type="submit">Add Expense</button>  </div>  </form>  );  };  export default ExpenseForm; |

We don't want to execute a function in the ExpenseForm component but in NewExpense instead. We will call the function stopEditingHandler where we call setIsEditing(false), which sets isEditing to false.

|  |
| --- |
| src/components/NewExpense/NewExpense.js |
| import React, {useState} from "react";  import ExpenseForm from "./ExpenseForm";  import "./NewExpense.css";  const NewExpense = (props) => {  const [isEditing, setIsEditing] = useState(false);  const saveExpenseDataHandler = (enteredExpenseData) => {  const expenseData = {  ...enteredExpenseData,  id: Math.random().toString(),  };  props.onAddExpense(expenseData);  };  const startEditingHandler = () => {  setIsEditing(true);  }  **const stopEditingHandler = () => {**  **setIsEditing(false);**  **}**  return (  <div className="new-expense">  {!isEditing && <button onClick={startEditingHandler}>Add New Expense</button>}  {isEditing && <ExpenseForm onSaveExpenseData={saveExpenseDataHandler} />}  </div>  );  };  export default NewExpense; |

The stopEditingHandler function is the function that we want to trigger which we in the end want to trigger if the "Cancel" button is pressed in the ExpenseForm component. We just need to pass a pointer at stopEditingHandler down to the ExpenseForm component. In the call to the ExpenseForm component within the NewExpense component, we add a new prop called onCancel and point at the stopEditingHandler function, so that the function stopEditingHandler function is passed as a value to the onCancel prop in the ExpenseForm component.

|  |
| --- |
| src/components/NewExpense/NewExpense.js |
| import React, {useState} from "react";  import ExpenseForm from "./ExpenseForm";  import "./NewExpense.css";  const NewExpense = (props) => {  const [isEditing, setIsEditing] = useState(false);  const saveExpenseDataHandler = (enteredExpenseData) => {  const expenseData = {  ...enteredExpenseData,  id: Math.random().toString(),  };  props.onAddExpense(expenseData);  };  const startEditingHandler = () => {  setIsEditing(true);  }  const stopEditingHandler = () => {  setIsEditing(false);  }  return (  <div className="new-expense">  {!isEditing && <button onClick={startEditingHandler}>Add New Expense</button>}  {isEditing &&  <ExpenseForm  onSaveExpenseData={saveExpenseDataHandler}  **onCancel={stopEditingHandler}** />}  </div>  );  };  export default NewExpense; |

Then, in the ExpenseForm component, we can now simply point at props.onCancel for a click on the "Cancel" button. So that the function onCancel, which we receive on the onCancel prop is forwarded to the onClick prop of this button, so that for a click on the button, the function stored in onCancel is executed, which of course is the stopEditingHandler function.

|  |
| --- |
| src/components/NewExpense/ExpenseForm.js |
| import React, { useState } from "react";  import "./ExpenseForm.css";  const ExpenseForm = (props) => {  const [enteredTitle, setEnteredTitle] = useState("");  const [enteredAmount, setEnteredAmount] = useState("");  const [enteredDate, setEnteredDate] = useState("");  // const [userInput, setUserInput] = useState({  // enteredTitle: '',  // enteredAmount: '',  // enteredDate: ''  // });  const titleChangeHandler = (event) => {  setEnteredTitle(event.target.value);  // setUserInput({  // ...userInput,  // enteredTitle: event.target.value,  // })  // setUserInput((prevState) => {  // return { ...prevState, enteredTitle: event.target.value}  // })  };  const amountChangeHandler = (event) => {  const dateParts = event.target.value.split("-");  setEnteredAmount(dateParts[0], dateParts[1] + 1, dateParts[2]);  // setUserInput({  // ...userInput,  // enteredAmount: event.target.value,  // })  };  const dateChangeHandler = (event) => {  setEnteredDate(event.target.value);  // setUserInput({  // ...userInput,  // enteredDate: event.target.value,  // })  };  // Handle date issue  const dateString = enteredDate;  const dateParts = dateString.split("-");  const aDate = new Date(dateParts[0], dateParts[1] - 1, dateParts[2]);  const submitHandler = (event) => {  event.preventDefault();  const expenseData = {  title: enteredTitle,  amount: enteredAmount,  date: aDate,  };  props.onSaveExpenseData(expenseData);  setEnteredTitle("");  setEnteredAmount("");  setEnteredDate("");  };  return (  <form onSubmit={submitHandler}>  <div className="new-expense\_\_controls">  <div className="new-expense\_\_control">  <label>Title</label>  <input  type="text"  value={enteredTitle}  onChange={titleChangeHandler}  />  </div>  <div className="new-expense\_\_control">  <label>Amount</label>  <input  type="number"  value={enteredAmount}  min="0.01"  step="0.01"  onChange={amountChangeHandler}  />  </div>  <div className="new-expense\_\_control">  <label>Date</label>  <input  type="date"  value={enteredDate}  min="2019-01-01"  max="2022-12-31"  onChange={dateChangeHandler}  />  </div>  </div>  <div className="new-expense\_\_actions">  <button type="button" **onClick={props.onCancel}**>Cancel</button>  <button type="submit">Add Expense</button>  </div>  </form>  );  };  export default ExpenseForm; |

We also want to close the form if it's submitted, so in saveExpenseDataHandler within the NewExpense component, we call setIsEditing(false);, which sets isEditing to false.

|  |
| --- |
| src/components/NewExpense/NewExpense.js |
| import React, {useState} from "react";  import ExpenseForm from "./ExpenseForm";  import "./NewExpense.css";  const NewExpense = (props) => {  const [isEditing, setIsEditing] = useState(false);  const saveExpenseDataHandler = (enteredExpenseData) => {  const expenseData = {  ...enteredExpenseData,  id: Math.random().toString(),  };  props.onAddExpense(expenseData);  **setIsEditing(false);**  };  const startEditingHandler = () => {  setIsEditing(true);  }  const stopEditingHandler = () => {  setIsEditing(false);  }  return (  <div className="new-expense">  {!isEditing && <button onClick={startEditingHandler}>Add New Expense</button>}  {isEditing &&  <ExpenseForm  onSaveExpenseData={saveExpenseDataHandler}  onCancel={stopEditingHandler} />}  </div>  );  };  export default NewExpense; |

We our now able to reload the page and have the form close when a new item is submitted. The form also closes when the "Cancel" button is clicked.

